NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

EXPLOITING CONSECUTIVE ONES STRUCTURE IN THE SET PARTITIONING PROBLEM

by

Mehmet Ayik

December 2000

Dissertation Supervisor: Gerald G. Brown

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY	2. REPORT DATE December 2000	3. REPORT TYPE AND DATES COVERED Dissertation	
4. TITLE AND SUBTITLE: Exploiting Consecutive Ones Structure in the Set Partitioning Problem		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Mehmet Ayik			
7. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**	

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT

The Set Partitioning Problem (SPP) is one of the most extensively researched models in integer optimization, and is widely applied in operations research. SPP is used for crew scheduling, vehicle routing, stock cutting, production scheduling, and many other combinatorial problems. The power and generality of SPP come at a price: An SPP can be very difficult to solve. A real-world SPP often has columns, or rows, with long strings of consecutive ones. We exploit this with a new preprocessing reduction that can eliminate some variables. We also introduce a column-splitting technique to render a model that can be solved directly or used to bound SPP with Lagrangian relaxation or an exterior penalty method. We develop an SPP row-splitting method that yields a special model that Bender's decomposition may then solve faster than the monolithic SPP. We demonstrate these techniques with well-known test problems from airlines and other researchers. We also contribute a new U.S. Navy aircraft carrier long-term deployment scheduling model, using our new techniques to plan with weekly fidelity over a ten-year planning horizon. This improved time fidelity increases planned deployment coverage of areas of responsibility by about ten carrier weeks.

14. SUBJECT TERMS Set Partitioning, Consecutive Ones, Preprocessing, Problem Size Reduction, Set Packing, Lagrangian Relaxation, Subgradient Optimization, Penalty Method, Benders Decomposition, Aircraft Carrier, Optimization	15. NUMBER OF PAGES 178
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

EXPLOITING CONSECUTIVE ONES STRUCTURE IN THE SET PARTITIONING PROBLEM

Mehmet Ayik
Lieutenant, Turkish Navy
B.S., Turkish Naval Academy, 1992
M.S., Naval Postgraduate School, 1998

Submitted in partial fulfillment of the
requirements for the degree of

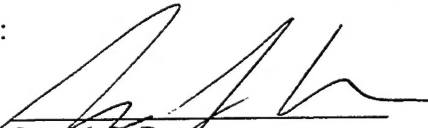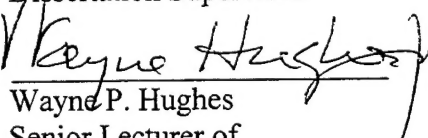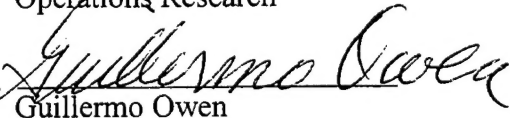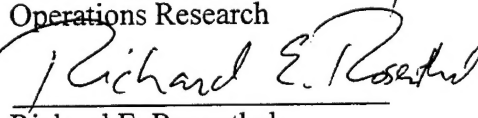## DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

from the
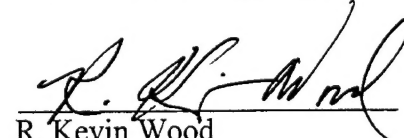
NAVAL POSTGRADUATE SCHOOL
December 2000
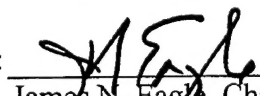
Author: _____
Mehmet Ayik
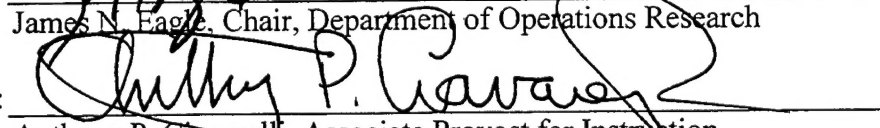
Approved by:

Gerald G. Brown
Professor of Operations Research
Dissertation Supervisor

Robert F. Dell
Associate Professor of
Operations Research

Wayne P. Hughes
Senior Lecturer of
Operations Research

Guillermo Owen
Professor of Mathematics

Richard E. Rosenthal
Professor of Operations Research

R. Kevin Wood
Professor of Operations Research

Approved by: _____
James N. Eagle, Chair, Department of Operations Research

Approved by: _____
Anthony P. Ciavarelli, Associate Provost for Instruction

iii

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Set Partitioning Problem (SPP) is one of the most extensively researched models in integer optimization, and is widely applied in operations research. SPP is used for crew scheduling, vehicle routing, stock cutting, production scheduling, and many other combinatorial problems. The power and generality of SPP come at a price: An SPP can be very difficult to solve. A real-world SPP often has columns, or rows, with long strings of consecutive ones. We exploit this with a new preprocessing reduction that can eliminate some variables. We also introduce a column-splitting technique to render a model that can be solved directly or used to bound SPP with Lagrangian relaxation or an exterior penalty method. We develop an SPP row-splitting method that yields a special model that Bender's decomposition may then solve faster than the monolithic SPP. We demonstrate these techniques with well-known test problems from airlines and other researchers. We also contribute a new U.S. Navy aircraft carrier long-term deployment scheduling model, using our new techniques to plan with weekly fidelity over a ten-year planning horizon. This improved time fidelity increases planned deployment coverage of areas of responsibility by about ten carrier weeks.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

THIS PAGE INTENTIONALLY LEFT BLANK

I.   INTRODUCTION

A.   BACKGROUND AND MOTIVATION

Among all special structures in pure integer programming, three have the most

widespread applications: set partitioning, set covering, and set packing.   Using set

notation, these problems can be expressed as follows.

Let $M = \{1,...,m\}$ and $N = \{1,...n\}$. Let M_j be a subset of M with an associated

weight of c_j, for all $j \in N$.   A subset S of N is a *cover* of M if $\bigcup_{j \in S} M_j = M$. S is a

packing of M if $M_j \bigcap M_k$ is empty for all $j,k \in S, j \neq k$. S is a *partition* of M if it is both

a cover and a packing of M.   The weight of a subset S of N is defined as $\sum_{j \in S} c_j$.   Figure I.1

illustrates a cover, a partition, and a packing of a set with six objects.

(a)   (b)   (c)
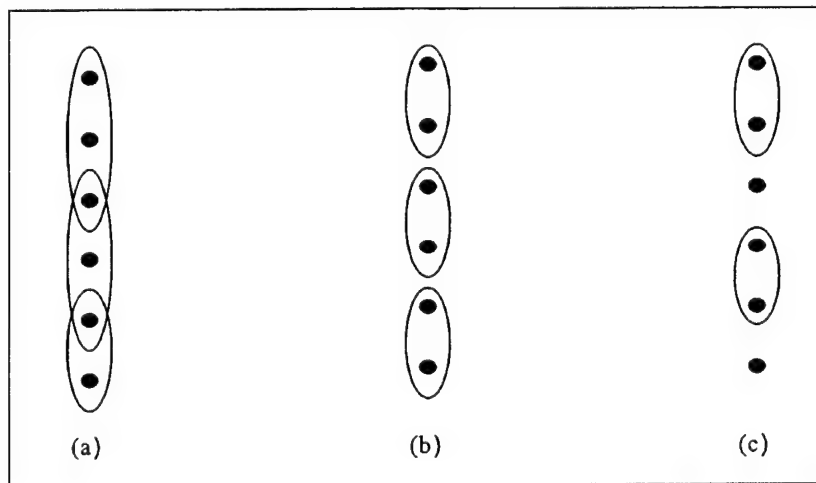
Figure I.1. A Cover, a Partition, and a Packing. Figure I.1.(a) is a cover, (b) is a
partition, and (c) is a packing of a set with six objects, say M. The collection of objects
covered by each oval border forms a subset of M. In Figure (a), each object in M is
covered at least once. The third and fifth ojects are covered twice. In (b), each object is
covered exactly once. In (c), the third and sixth objects are not covered and the two
subsets are disjoint.

In the set covering problem, the objective is finding a cover S of M with the minimum weight, whereas in the set packing problem, the objective is finding a packing S with the maximum weight. For the set partitioning problem both minimization and maximization versions are possible.

To formulate these problems as integer programming problems, we introduce the $m \times n$ *incidence matrix* A of the family $\{M_j \mid j \in N\}$, whose entries are given by $a_{ij} = 1$ if $i \in M_j$, and $a_{ij} = 0$ otherwise. We also define a decision variable $x_j, j = 1,...,n,$ that is equal to 1 if $j \in S$, and 0 otherwise. Let $x = (x_1,...,x_n)$. Then S is a cover, pack, or partition if and only if, respectively:

$$Ax \geq e, Ax \leq e, \text{or} Ax = e,$$

where e is a column vector of size m consisting of all ones. In a more general case in which e has all its entries equal to a scalar k, S is named *k-cover*, *k-pack*, or *k-partition*, respectively.

The set partitioning, set packing, and set covering problems are proven to be NP-Complete ([Lenstra and Rinnooy Kan 1979], [Garey and Johnson 1979]).

In the following subsections, we provide motivation for the set partitioning, set packing and set covering problems with special emphasis on a variety of subordinate topics including: problem formulations, applications, algorithms, and the inter-relationships between these problems.

1.   Set Partitioning Problem (SPP)

The integer programming formulation of SPP is:

$$\text{minimize} \quad c'x \tag{I.1.a}$$

$$(SPP) \quad \text{s.t.} \quad Ax = e \tag{I.1.b}$$

$$x \text{ binary} \tag{I.1.c}$$

where c is a column vector of size n and c' denotes its transpose. Using standard notation

(e.g., [Bertsimas and Tsitsiklis 1997]), column j of matrix A is denoted as A_j, and row i

of A is denoted as a_i'. In Equation (I.1.b), Ax is referred to as the *left-hand side* of the

equality, and e as the *right-hand side*. All vectors are assumed to be column vectors.

a.   *Applications Of SPP*

A wide variety of practical applications have been modeled as SPPs. A

partial list of applications described in the literature includes: *crew scheduling* ([Charnes

and Miller 1956], [Marsten and Shepardson 1980], [Hoffman and Padberg 1993]), *vehicle

routing* ([Brown et al. 1987a]), *stock cutting* ([Pierce 1970]), *political districting*

([Garfinkel and Nemhauser 1970]), and *circuit partitioning* ([Eben-Chaime et al. 1996]).

References to further applications can be found in [Garfinkel and Nemhauser 1972,

Chapter 8], [Balas and Padberg 1976], and [El-Darzi and Mitra 1990].

Well-known military applications of SPP include a variety of ship

scheduling problems.   Wing [1986] schedules surface combatants for inspections,

training, and other events. Brown et al. [1990] schedule U.S. Atlantic Fleet combatants

to deployments and naval exercises. Ayik [1998] presents a set partitioning model, also

3

involving set packing and set covering constraints, to schedule the U.S. aircraft carriers for deployment and maintenance.

The best-known application of SPP is airline crew scheduling. The costs associated with assigning personnel to flights are the second highest operating expenditure in the airline industry [Hoffman and Padberg 1993], hence the financial significance of crew scheduling.

b. Solution Algorithms For SPP

Linear programming (LP) based *branch and bound* (*B&B*) (e.g., Bertsimas and Tsitsiklis 1997]) is the most common approach used to solve an integer programming problem. B&B is used as an infrastructure in most of the efficient SPP algorithms.

The most frequently published approaches for solving SPP other than by outright B&B are *implicit enumeration* and *cutting plane methods*. Balas and Padberg [1976] provide a survey of these and other approaches. Implicit enumeration takes advantage of the special structure of SPP. Systematic search of the solution space generates partial solutions (assigning zero-one values to variables taken one at a time) and explores the logical implications of these value assignments.

Hoffman and Padberg [1993] present a *branch-and-cut*[†] approach to solve large scale SPPs. A branch-and-cut solver generates cutting planes based on the underlying structure of the *polytope*[†] defined by the *convex hull*[†] of the feasible integer

---

[†]This terminology is defined in basic texts such as Bertsimas and Tsitsiklis [1997], and Nemhauser and Wolsey [1988].

points, and incorporates these cuts into a B&B tree-search that uses automatic reformulation procedures, heuristics and LP technology to assist in the solution. There are four components to a branch-and-cut optimizer: a preprocessor that tightens the user-supplied formulation; a heuristic that yields good integer feasible solutions quickly; a cut generation procedure (the engine of this overall approach that tightens the LP relaxation), and a branching strategy that selects the next branching variable and determines the search-tree. Hoffman and Padberg claim to be very efficient at solving SPPs.

c. *Problem Size Reduction*

Problem size reduction techniques are implemented on SPP to reduce the number of variables and/or constraints through logical implications without eliminating optimal solutions to the original problem. Problem size reduction, also called presolve or prereduce, is an effective and inexpensive tool used by all efficient SPP algorithms today.

Hoffman and Padberg's branch-and-cut optimizer uses size reduction before at each node of the branch-tree that is associated with a non-trivial problem restriction, say fixing a binary variable to one. The idea is to propagate and amplify the effects of variable fixing based on reduced costs and branching decisions. They report that these techniques are highly effective in reducing the solution times of the LP subproblems within branch-and-cut.

Problem reductions are typically based on recognition of duplicate columns, redundant rows, and conflicting variables ([Balas and Padberg 1976], and [Hoffman and Padberg 1993]). Ali et al. [1995] present variable reductions based on hidden network structure in SPP.

5

Problem reductions are discussed in Chapter V.

d.   *Use Of Special Structures*

The identification of special structures within the incidence matrix, A, can play a central role in the solution procedures for SPP. In LP-based B&B of SPP, special structures are used to aid in fathoming and branching. For instance, to aid in branching, Marsten [1974] uses a reordering of the columns and rows, and Avis [1980] identifies dominance relations in the incidence matrix, A.

Special structures, either inherent or enforced (artificially extracted), are also used in SPP solution techniques. Embedded structure in the incidence matrix, such as generalized upper bounds, network rows, and generalized networks can be recognized with very little effort (e.g., [Brown and Thomen 1980], [Brown and Wright 1984], [Brown et al. 1985]). Nemhauser and Weber [1979] enforce a bipartite matching to solve the large-scale LP relaxations of SPPs (with an associated increase in the number of variables and constraints).

Marsten and Shepardson [1980] present *column splitting* to reveal the network structure of the *two-duty period scheduling* problem. (This technique is described in Chapter II.) The two-duty scheduling problem, arising naturally in personnel scheduling, is formulated as an SPP. The formulation of the problem is described as follows.

Suppose there are a number of duty stations, each of which has minimum staffing requirements at every period of the working day. The rows of the incidence matrix, A, correspond to the hours of operation of each station. Note that the rows are in

6

natural order, arranged by station and ordered sequentially by time periods within each station. Each column corresponds to a possible personnel schedule where an entry of 1 in the matrix indicates that the column's worker is assigned to the station of that row for the corresponding time period. With the restrictions that a worker is assigned to no more than one station during his morning duty period and no more than one (possibly different) station during his afternoon duty period, each column will contain at most two strings of ones in consecutive matrix rows. The problem is to find a minimal-cost set of personnel schedules such that each station's duty requirements will be satisfied.

In the case where each person is allowed to work exactly one duty period a day (i.e., each column of the incidence matrix has one string of consecutive ones), the problem is called a *one-duty period scheduling* problem. The one-duty period scheduling problem can be transformed to a minimum cost network flow model, and thus be solved in *polynomial time* (e.g., [Veinott and Wagner 1962], [Garfinkel and Nemhauser 1972]). That is, the time needed to solve the problem is a polynomial function of the length of the input data.

Circular ones in all columns permits SPP to be solved parametrically as a bounded series of network flow problems (e.g., [Bartholdi et al. 1980]). A 0-1 column is said to be circular if its ones occur consecutively, where the last and first entries are also considered to be consecutive.

Ali and Thiagarajan [1989], and Ali et al. [1995] use hidden network structures in SPP to transform the problem to a network with side constraints and side columns, respectively.

2. Set Packing (SP) And Set Covering (SC)

The integer programming formulations of SP and SC problems are given by (*SP*) and (*SC*), respectively:

$$\text{maximize} \quad c'x \qquad \text{(I.2.a)}$$

$$(SP) \quad \text{s.t.} \quad Ax \leq e \qquad \text{(I.2.b)}$$

$$x \text{ binary} \qquad \text{(I.2.c)}$$

$$\text{minimize} \quad c'x \qquad \text{(I.3.a)}$$

$$(SC) \quad \text{s.t.} \quad Ax \geq e \qquad \text{(I.3.b)}$$

$$x \text{ binary} \qquad \text{(I.3.c)}$$

SP and SC problems are close relatives of SPP. An SPP can be formulated as an SP or SC problem (e.g., [Balas and Padberg 1976]). Conversely, an SP problem can be formulated as an SPP. However, we cannot formulate an SC problem as an equivalent SPP.

The equivalence relationships between SP, SC, and SPP show that the applications referenced in the previous section for SPP can also be listed for SP or SC problems. For a partial list of SC problem specific applications, the reader is referred to [Beasley 1987], [Fisher and Kedia 1990], and [Grossman and Wool 1997].

There is abundant literature on the SC problem, dealing with: *exact algorithms* (e.g., [Beasley 1992], and [Fisher and Kedia 1990]), *heuristics* (e.g., [Beasley and Chu 1996], [Haddadi 1997], and [Caprara et al. 1999]), and *surveys* ([Garfinkel and Nemhauser 1972] and [Christofides and Korman 1975]).

8

Caprara et al. [1999] present a Lagrangian-based heuristic for the SC problem that we adapt for SPP. The algorithm is designed to solve large-scale SC problem instances with up to 5,500 rows and 1,100,000 columns, arising from crew scheduling for an Italian railway. The primary characteristics of the algorithm include: (1) a dynamic pricing scheme for the variables, similar to that used for solving large-scale linear programs, coupled with subgradient optimization and greedy heuristics, and (2) the systematic use of column fixing to obtain improved solutions. Additionally, Caprara et al. present several improvements on the standard way of defining the step-size and the ascent direction within the *subgradient optimization* procedure. (For comprehensive information on the Lagrangian relaxation problem and the subgradient optimization method see [Parker and Rardin 1988]). Caprara et al. report this algorithm to be more efficient than existing heuristics.

3. A Long-Term Aircraft Carrier Deployment Problem Incorporating Set Partitioning, Set Packing, And Set Covering Constraints

A United States (U.S.) Navy aircraft carrier scheduling problem can be formulated using a classical set partitioning model that also involves set covering and set packing constraints ([Ayik 1998]). Many researchers have formulated the scheduling of transportation vehicles (e.g., delivery trucks, buses, oil tankers and ships) as an SC or SPP. Appelgren [1969, 1971] and Crawford and Sinclair [1977] suggest SC or SPP to respectively schedule ships and tankers. Brown et al. [1987a] schedule crude oil super tankers using a set partitioning formulation. Military applications of this approach

include the scheduling of the U.S. Navy combatants to deployments and naval exercises (e.g., [Wing 1986], [Brown et al. 1990]).

The classical set partitioning approach first generates all possible schedules that provide the period-by-period status of each carrier for the planning horizon while satisfying operations and maintenance constraints. Next, an SPP is formulated to maximize coverage (or minimizing uncovered periods) in areas of responsibility (AORs) subject to the constraints that

(i)   exactly one alternate schedule is chosen for each carrier, and

(ii)   each AOR should be covered in each period.

The algebraic formulation of the set partitioning model is as follows:

Indices:

c carriers

a Areas of Responsibility (AORs)

t periods (in weeks)

$j \in J(c)$ set of *possible schedules* for each carrier c (i.e. schedules that satisfy the operations and maintenance constraints, and provide the period-by-period status of this carrier for the planning horizon)

Data:

A_{ctj}^{a} equals 1 if schedule j of carrier c covers AOR a in period t, 0 otherwise

$WEIGHT^{a}$ weight of coverage in AOR a

$MAXGAP$ maximum allowable number of consecutive uncovered periods in an AOR

Decision Variables

x_j equals 1 if schedule j is selected, 0 otherwise

$uncovered_t^{a}$ equals 1 if AOR a is not covered in period t, 0 otherwise

Formulation

$$\text{minimize} \quad \sum_{a,j} WEIGHT^{a} uncovered_j^{a} \qquad \text{(I.4.a)}$$

$$\text{s.t.} \quad \sum_{j \in J(c)} x_j = 1 \qquad \forall \; c \qquad \text{(I.4.b)}$$

$$\sum_{c,j} A_{ctj}^{a} x_j + uncovered_t^{a} \geq 1 \qquad \forall \; a, t \qquad \text{(I.4.c)}$$

$$\sum_{t'=t}^{t+(MAXGAP-1)} uncovered_{t'}^{a} \leq (MAXGAP - 1) \qquad \forall \; a, t \qquad \text{(I.4.d)}$$

$$x_j \in \{0,1\} \qquad \forall \; j \in J(c), c \qquad \text{(I.4.e)}$$

$$uncovered_t^{a} \geq 0 \qquad \forall \; a, t \qquad \text{(I.4.f)}$$

11

In the above formulation, the objective is to minimize the uncovered periods in all AORs. Partition constraints (I.4.b) ensure that exactly one schedule is selected for each carrier. Constraints (I.4.c) express the intent that each AOR should be covered in each period. Because covering all AORs is not possible with the current carrier force, this constraint has an elastic variable for each uncovered period. Packing constraints (I.4.d) ensure that uncovered periods for each AOR are no more than the maximum allowable number of gap periods (*MAXGAP*).

Table I.1 shows the size of this SPP for twelve aircraft carriers, fixed maintenance periods, weekly time increments, and a planning horizon of ten years. The number of columns increases to several million if the maintenance periods are scheduled synchronously with the deployment periods.

Number of Constraints			Number of Variables	
Partitioning	Covering	Packing	x	*uncovered*
14	1,046	1,046	222,293	1,046

Table I.1. Model Size for the Set Partitioning Formulation of the U.S. Navy Aircraft Carrier Deployment Scheduling Problem with Twelve Aircraft Carriers, and Fixed Maintenance Periods, Weekly Time Periods, and a Ten-Year Planning Horizon.

Set partitioning is attractive in ship scheduling because it is relatively easy to generate, modify, and control. Although set partitioning has many advantages, it has not been the preferred method for solving the carrier scheduling problem because the long planning horizon yields an impractically large number of alternate schedules.

B.   OUTLINE OF THE DISSERTATION

A real-world SPP often has columns, or rows, with long strings of consecutive ones. This dissertation seeks methods to bound, or solve an SPP by exploiting the consecutive ones structure in the incidence matrix, A.

Chapter II presents some of the preliminaries that are applied throughout this dissertation, including:

(i)   a column splitting reformulation of SPP,

(ii)   total unimodularity, and

(iii)   Lagrangian relaxation.

Chapter III presents methods to solve the column split SPP reformulated problem. We focus on finding a good lower bound that can be incorporated in branch-and-cut [Hoffman and Padberg 1993] to solve the original SPP.

In Chapter IV, we present a new algorithm to solve binary programming problems (e.g., SP, SC, or SPP) whose rows contain strings, or *segments*, of consecutive ones. This algorithm can also solve general binary programming problems. However, the runtime of the algorithm degrades as the number of ones segments in the problem increases.

Chapter V discusses problem size reduction in SPP. We first present the known reduction techniques.   Then, we show other reductions suggested by the network constraints obtained by the reformulation of SPP using column splitting.

Chapter VI presents a new integer programming formulation for the aircraft carrier scheduling problem. We first describe the scheduling factors and operations constraints. Next, we present the previously suggested model, a two-commodity network

13

flow problem with side constraints. Then, we introduce the new formulation and compare it with the previous models.

Finally, Chapter VII concludes this dissertation by summarizing the primary findings, and offering suggestions for future research.

II. PRELIMINARIES

A column split reformulation of SPP proceeds as follows. Consider each existing column in the SPP coefficient matrix A. Identify each segment of rows with consecutive ones in this column, and define a corresponding new binary column in the split reformulation with these same consecutive ones. Also add coupling constraints to the reformulation that all the new split binary columns associated with each original SPP column must share the same value. This column split reformulation exhibits purely consecutive ones columns in the rows it inherits from the seminal SPP, and thus is totally unimodular in these original rows. In addition, the new coupling constraint rows are trivially constructed with total unimodulatity. However, although the original rows are now unimodular, and the new rows are constructed to be unimodular, the union of these two individually unimodular sets of rows is not unimodular. But, by moving the coupling constraints to the objective function and implementing a well-known Lagrangian relaxation procedure, we leave ourselves with the unimodular restatement of the original constraints and can obtain lower bounds for SPP.

This chapter demonstrates the column splitting reformulation of SPP, reviews total unimodularity, and Lagrangian relaxation to help us construct the methods presented in the following chapters.

A.   COLUMN SPLITTING TECHNIQUE

Marsten and Shepardson [1980] first introduce column (variable) splitting (decoupling) to solve the two-duty period scheduling problem. Marsten and Shepardson

use this technique to reformulate the two-duty period scheduling problem as a network flow problem with side constraints.

Consider the following SPP:

$$\text{minimize} \quad \sum_{j=1}^{n} c_j x_j \qquad \qquad \text{(II.1.a)}$$

$$(SPP) \quad \text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j = 1 \qquad i = 1,...,m \qquad \qquad \text{(II.1.b)}$$

$$x_j \; binary \qquad j = 1,...,n \qquad \qquad \text{(II.1.c)}$$

where c_j represents the cost of variable x_j for all $j = 1,...,n$, and the scalar $a_{ij} \in \{0,1\}$ for all $i = 1,...,m$ and $j = 1,...,n$.

In (SPP), for each column vector A_j we define a *segment of ones* as a consecutive set of column elements a_{ij}, $i = l_j, l_j + 1,..., l_j + p_j - 1, l_j + p_j$, such that $a_{ij} = 1$ for $i = l_j,..., l_j + p_j$, $a_{l_j - 1, j} = 0$ if $l_j > 1$, and $a_{l_j + p_j + 1, j} = 0$ if $l_j + p_j < m$. Let K_j be the set of segments of ones for column j containing the information where each segment starts and ends in terms of pairs of ordered row indices (e.g., $K_j = \{(l_j, l_j + p_j), ...\}$). $|K_j|$ denotes the number of segments in column j. Let Γ be the set of columns that have more than one segment of ones (i.e., $\Gamma = \{j : |K_j| > 1\}$). Let A_j^k be a column vector of size m, such that $a_{ij}^k = 1$ if the ith element of column j is in the kth segment of K_j in (SPP), and $a_{ij}^k = 0$ otherwise. Note that the sum of A_j^k for all segments in K_j is equal to

16

the vector A_j (i.e., $\sum_{k=1}^{|K_j|} A_j^k = A_j$). Column A_j^k is given a cost coefficient of $d_j^k = c_j \alpha_j^k$

where $\sum_{k=1}^{|K_j|} \alpha_j^k = 1$, and is associated with the variable $y_j^k \in \{0,1\}$. The y_j^k are required to

satisfy $y_j^k = y_j^{k+1}$ for all $k = 1,...,|K_j|-1$ and $j \in \Gamma$. Thus, we obtain a problem

equivalent to (SPP) given by:

$$\text{minimize} \quad \sum_{j=1}^{n} \sum_{k=1}^{|K_j|} d_j^k y_j^k \tag{II.2.a}$$

$$(SPP') \quad \text{s.t.} \quad \sum_{j=1}^{n} \sum_{k=1}^{|K_j|} a_{ij}^k y_j^k = 1 \qquad \forall\, i \tag{II.2.b}$$

$$y_j^k - y_j^{k+1} = 0 \qquad j \in \Gamma \text{ and } k = 1,...,|K_j|-1 \tag{II.2.c}$$

$$y_j^k \text{ binary} \qquad \forall\, j \text{ and } k \tag{II.2.d}$$

Using matrix notation, this problem can be written as:

$$\text{minimize} \quad d'y \tag{II.3.a}$$

$$(SPP') \quad \text{s.t.} \quad Ty = e \tag{II.3.b}$$

$$Sy = 0 \tag{II.3.c}$$

$$y \text{ binary} \tag{II.3.d}$$

where T and S are the coefficient matrices associated with equations (II.2.b) and (II.2.c),

respectively. By construction, S has the node-arc incidence matrix structure of a network

(i.e., each column of S has either a +1 and a −1, only a +1, only a −1, or all zeros), and T

has exactly one segment of ones in each column.

17

Example II.1 illustrates the reformulation described thus far.

Example II.1: Consider the following SPP where $m = n = 6$.

$$\text{minimize} \quad c'x$$

(SPP)   s.t. $Ax = e$

$$x \text{ binary}$$

$$c = \begin{bmatrix} 4 & 2 & 6 & 8 & 3 & 4 \end{bmatrix}$$

$$
A = \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array}
\begin{array}{cccccc}
A_1 & A_2 & A_3 & A_4 & A_5 & A_6 \\
1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1
\end{array}
$$

We split the columns of A to obtain one segment of ones in each column. T is the resulting matrix.

$$
T =
\begin{array}{ccccccccccccc}
A_1^1 & A_1^2 & A_2^1 & A_2^2 & A_3^1 & A_3^2 & A_4^1 & A_4^2 & A_5^1 & A_5^2 & A_5^3 & A_6^1 & A_6^2 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1
\end{array}
$$

Distributing the cost equally among the progeny of each split column, we obtain the following cost coefficient vector d:

$$
\begin{array}{ccccccccccccc}
A_1^1 & A_1^2 & A_2^1 & A_2^2 & A_3^1 & A_3^2 & A_4^1 & A_4^2 & A_5^1 & A_5^2 & A_5^3 & A_6^1 & A_6^2 \\
\end{array}
$$
$$d = \begin{bmatrix} 2 & 2 & 1 & 1 & 3 & 3 & 4 & 4 & 1 & 1 & 1 & 2 & 2 \end{bmatrix}$$

Thus, the equivalent reformulation is as follows:

$$\min \quad 2y_1^1 + 2y_1^2 + y_2^1 + y_2^2 + 3y_3^1 + 3y_3^2 + 4y_4^1 + 4y_4^2 + y_5^1 + y_5^2 + y_5^3 + 2y_6^1 + 2y_6^2$$

$$
\begin{aligned}
\text{s.t.} \quad y_1^1 + y_3^1 + y_5^1 &= 1 \\
y_1^1 + y_4^1 + y_6^1 &= 1 \\
y_1^1 + y_2^1 + y_3^2 + y_6^1 &= 1 \\
y_3^2 + y_5^2 &= 1 \\
y_2^2 + y_3^2 &= 1 \\
y_1^2 + y_2^2 + y_4^2 + y_5^3 + y_6^2 &= 1 \\
\hline
y_1^1 - y_1^2 &= 0 \\
y_2^1 - y_2^2 &= 0 \\
y_3^1 - y_3^2 &= 0 \\
y_4^1 - y_4^2 &= 0 \\
y_5^1 - y_5^2 &= 0 \\
y_5^2 - y_5^3 &= 0 \\
y_6^1 - y_6^2 &= 0
\end{aligned}
$$

$$y_1^1,\ y_1^2,\ y_2^1,\ y_2^2,\ y_3^1,\ y_3^2,\ y_4^1,\ y_4^2,\ y_5^1,\ y_5^2,\ y_5^3,\ y_6^1,\ y_6^2 \in \{0,1\}$$

End of Example II.1

Matrix T of formulation (SPP') has exactly one segment of consecutive ones in each column. A matrix having this structure is called an *interval matrix* (e.g., [Nemhauser and Wolsey 1988]). T can be transformed to a node-arc incidence matrix of a network (e.g., [Veinott and Wagner 1962]). To accomplish this transformation, we first append a redundant $(m+1^{st})$ constraint $0x = 0$ to the end of equation set (II.3.b). We next perform an elementary row operation for each $i = m+1, m, ..., 1$, subtracting the ith constraint in (II.3.b) from the $(i+1^{st})$ constraint. These operations create the formulation below:

$$\text{minimize} \quad d'y \qquad \qquad \text{(II.4.a)}$$

$$(NS) \qquad \text{s.t.} \qquad \mathcal{N}y = b \qquad \qquad \text{(II.4.b)}$$

$$Sy = 0 \qquad \qquad \text{(II.4.c)}$$

$$y \text{ binary} \qquad \qquad \text{(II.4.d)}$$

$\mathcal{N}$ is the resulting network matrix consisting of exactly one $+1$, and one -1 in each column. The number of rows of $\mathcal{N}$ is $(m+1)$. Column vector b has $+1$ as its first entry, is followed by zeroes, and has -1 as the last $(m+1^{st})$ entry.

If we consider constraints $Sy = 0$ as side constraints, then (NS) can be defined as a *constrained shortest path problem*. Each row of constraints $\mathcal{N}y = b$ corresponds to a node in the network. Every variable y_j^k is represented by an arc directed from a node in which y_j^k has a $+1$ coefficient, to a node with a -1 coefficient. Hence, the network corresponding to (NS) is called a *directed* network.

Cost coefficient d_j of variable y_j^k is assigned as the length of the arc corresponding to y_j^k. One unit of flow is sent from the first node $(i = 1)$ to the last node $(i = m+1)$. By construction, every arc in the network is *forward* (i.e., if an arc is incident from node i_1 to node i_2, then $i_1 < i_2$). Hence, the network corresponding to (NS) is *acyclic*. Side constraints $Sy = 0$ ensure that if we use arc y_j^l in the network, then we also use arc y_j^k for all $k \in \{1,...,|K_j|\} \setminus l$.

20

Observe that the shortest path of the network corresponding to (*NS*) is a lower bound on (*SPP*). Further, the longest path of the same network is an upper bound on the objective value of (*SPP*).

Example II.2 illustrates the reformulation (*NS*) for the set partitioning problem presented in Example II.1.

Example II.2:

By appending a row of zeros to matrix T in Example II.1, and subtracting the ith row from the $(i+1^{st})$ row, we obtain the following matrix, N.

$$
\mathcal{N} = \begin{array}{ccccccccccccc}
A_1^1 & A_1^2 & A_2^1 & A_2^2 & A_3^1 & A_3^2 & A_4^1 & A_4^2 & A_5^1 & A_5^2 & A_5^3 & A_6^1 & A_6^2 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & -1 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 \\
\end{array}
$$

Thus, the equivalent reformulation (NS) is:

$$\min \quad 2y_1^1 + 2y_1^2 + y_2^1 + y_2^2 + 3y_3^1 + 3y_3^2 + 4y_4^1 + 4y_4^2 + y_5^1 + y_5^2 + y_5^3 + 2y_6^1 + 2y_6^2$$

$$
\begin{aligned}
\text{s.t.} \quad & +y_1^1 & & & +y_3^1 & & & +y_5^1 & & & & = 1 \\
& & & -y_3^1 & +y_4^1 & -y_5^1 & & & +y_6^1 & & & = 0 \\
& +y_2^1 & & +y_3^2 & -y_4^1 & & & & & & = 0 \\
-y_1^1 & -y_2^1 & & & & +y_5^2 & & -y_6^1 & & & = 0 \\
& +y_2^2 & & & & -y_5^2 & & & & = 0 \\
& +y_1^2 & -y_3^2 & +y_4^2 & +y_5^3 & +y_6^2 & = 0 \\
-y_1^2 & -y_2^2 & & -y_4^2 & & -y_5^3 & -y_6^2 & = -1 \\
\end{aligned}
$$

$$
\begin{aligned}
y_1^1 - y_1^2 &= 0 \\
+y_2^1 - y_2^2 &= 0 \\
+y_3^1 - y_3^2 &= 0 \\
+y_4^1 - y_4^2 &= 0 \\
+y_5^1 - y_5^2 &= 0 \\
+y_5^2 - y_5^3 &= 0 \\
+y_6^1 - y_6^2 &= 0 \\
\end{aligned}
$$

$$y_1^1, \ y_1^2, \ y_2^1, \ y_2^2, \ y_3^1, \ y_3^2, \ y_4^1, \ y_4^2, \ y_5^1, \ y_5^2, \ y_5^3, \ y_6^1, \ y_6^2 \in \{0,1\}$$

End of Example II.2

Proposition II.1 is a result of the operations shown thus far.

Proposition II.1: $y_j^k{}^*$ is an optimal solution of (NS) if and only if $x_j^* = y_j^1{}^*$ is an optimal solution to (SPP) for all $j = 1, ..., n$.

Different equivalent reformulations of (SPP) can also be derived using the column splitting technique. Side constraints (II.4.c) may be formulated as any equivalent set of equalities that together imply that $y_j^1 = y_j^2 = ... = y_j^{|K_j|-1} = y_j^{|K_j|}$. For instance:

- $y_j^r = y_j^k$ $\qquad\qquad\qquad\qquad \forall\; k \in \left(\left\{ 1,...,\left|K_j\right| \right\} \setminus r \right)$ and $j \in \Gamma$ $\qquad$ (II.5.a)

- $y_j^r = y_j^k$ $\qquad \forall\; r = 1,...,\left|K_j\right|,\; k = 1,...,\left|K_j\right|$ such that $r \neq k$ and $j \in \Gamma$ $\qquad$ (II.5.b)

- $\left(\left|K_j\right| - 1\right)y_j^r - \displaystyle\sum_{k \in \left(\left\{1,...,\left|K_j\right|\right\}\setminus r\right)} y_j^k = 0$ $\qquad\qquad\qquad \forall\; j \in \Gamma$ $\qquad$ (II.5.c)

Note that constraints (II.5.c) do not yield as strong of an LP relaxation as the other equivalent reformulations of SPP.

The arguments stated in this section for the reformulation of SPP also hold for SC and SP with minor adjustments. Moreover, any linear program can be reformulated using the column splitting technique to obtain special structures in each split column. For instance, Schrage [1997] shows that any linear program can be converted to one with no more than three non-zero coefficients per column, and we can use column splitting to render this into a generalized network with side constraints.

Matrices $\mathcal{N}$ and S of reformulation (NS) both have the node-arc incidence matrix structure of a network. Next, we show that the matrices with this structure are members of a class called totally unimodular (TU) Matrices.

B. TOTAL UNIMODULARITY

First, we define TU and list some of the well-known properties of TU matrices that are used throughout this dissertation. For comprehensive information on total unimodularity and related theorems, see [Nemhauser and Wolsey 1988].

23

Definition II.1: An $(m \times n)$ integral matrix A is totally unimodular if the determinant of each square submatrix of A is equal to 0, 1, or -1.

It is evident that $a_{ij} = 0$, 1, or -1 if A is TU, because every entry of the matrix is a (1×1) square submatrix.

Proposition II.3: An $(m \times n)$ matrix A is TU if and only if the matrix (A, I_m) is TU (where I_m is an $(m \times m)$ identity matrix).

Proposition II.4: A matrix A is TU if and only if the transpose matrix A' is TU.

Next, we present a significant theorem of integer programming developed by Hoffman and Kruskal [1956].

Theorem II.5: If A is TU, and if $\bar{b}$, $\underline{b}$, l, and u are integral, then every basic feasible solution defined by the constraints $\underline{b} \leq Ax \leq \bar{b}$, $l \leq x \leq u$ is integral.

Proposition II.6: The node-arc incidence matrix N of a directed network is TU.

Thus, we complete our brief tour of the relevant properties of TU matrices and observe that matrices $\mathcal{N}$ and S of reformulation (SPP'') are TU.

C.   LAGRANGIAN RELAXATION

Lagrangian relaxation dates to the eighteenth century. More recent use of this method in discrete optimization appears in the seminal papers by Held and Karp [1970, 1971] that address the "traveling salesman problem." Fisher [1981, 1985], Geoffrion [1974], and Shapiro [1979] provide insightful surveys of the Lagrangian relaxation and its uses in integer programming.

Lagrangian relaxation is often used for integer programming problems

$$(IP) \quad z^* = \min\{c'x : Ax = b, \ x \text{ integral}\}$$

for which the constraints $Ax = b$ can be split into two parts, $A_1x = b_1$ and $A_2x = b_2$ such that *relaxed* problems of the form $\min\{c'x : A_1x = b_1, \ x \text{ integral}\}$ can be solved efficiently.

The Lagrangian relaxation method uses the idea of relaxing the explicit linear constraints by bringing them into the objective function with associated *Lagrange multipliers* μ. The resulting problem

$$\underset{x}{\text{minimize}} \quad c'x + \mu(A_2x - b_2)$$

$$\text{s.t.} \quad A_1x = b_1$$

$$x \text{ integral}$$

is referred to as a *Lagrangian relaxation* or *Lagrangian subproblem* of the original problem (*IP*), and the function

$$L(\mu) = \min_x \{c'x + \mu(A_2x - b_2) : A_1x = b_1, \ x \text{ integral}\}$$

is referred to as *Lagrangian function*. The solution of the Lagrangian subproblem need not be feasible for the original problem.

The Lagrangian relaxation method is motivated by the following observation:

Theorem II.7 (Lagrangian Bounding Principle): For any vector μ of Lagrange multipliers, the value $L(\mu)$ of the Lagrangian function is a lower bound on the optimal objective function value z^* of the original optimization problem (*IP*). (e.g., Ahuja et al. [1993, Chapter 16, pp. 605-606])

25

To obtain the highest possible lower bound, we need to solve the following optimization problem

$$L^* = \max_{\mu} L(\mu)$$

which is referred to as the *Lagrangian multiplier problem*.

Most of the key results of Lagrangian relaxation (e.g., the bounding principle and optimality conditions) are special cases of more general results in mathematical programming duality theory. Rockafellar [1970] and Stoer and Witzgall [1970] provide comprehensive treatments of this subject.

The preceding discussion of the Lagrangian bounding principle provides us with valid bounds for comparing objective function values of the Lagrangian multiplier problem and the original problem (*IP*) for any choices of the Lagrangian multipliers μ, and any feasible solution x of (*IP*):

$$L(\mu) \leq L^* \leq z^* \leq c'x.$$

Hence, the Lagrangian bounding principle has the following implication:

Corollary II.8: If $L(\mu) = c'x$ for some Lagrangian multiplier vector μ, and for a feasible solution x of (*IP*), then $L(\mu) = L^* = z^* = c'x$.

Furthermore, the following proposition defines more explicit bounds for the case in which the Lagrangian subproblem yields intrinsically integer solutions.

Proposition II.9: If the Lagrangian subproblem yields intrinsically integer solutions, then the optimal value L^* of the Lagrangian multiplier problem is equal to the optimal objective function value of the LP relaxation of (*IP*) [Geoffrion 1974].

26

III. SPP LOWER BOUND ALGORITHMS IMPLEMENTED WITH THE COLUMN SPLITTING REFORMULATION

This chapter presents two algorithms to solve the column splitting reformulation problem (NS):

    (i)   a Lagrangian relaxation method using subgradient optimization, and

    (ii)   an exterior penalty method.

We seek a good lower bound that can be computed with less effort than an LP relaxation of (SPP) and could be incorporated in branch-and-cut [Hoffman and Padberg 1993] to solve the original SPP. We also investigate the reordering of rows to reduce the number of segments of consecutive ones in the columns of an SPP.

A.   LAGRANGIAN RELAXATION AND SUBGRADIENT OPTIMIZATION

Marsten and Shephardson [1980] form a Lagrangian relaxation on the reformulated problem and use subgradient optimization. Marsten and Shepardson report their computational experience on the two-duty problem and suggest further investigation on three- or four-duty period problems.

Consider the reformulated problem (NS).

$$\text{minimize} \quad d'y$$

$$(NS) \quad \text{s.t.} \quad \mathcal{N}y = b$$

$$Sy = 0$$

$$y \ binary$$

By moving constraints $Sy = 0$ to the objective function, we obtain the following Lagrangian subproblem:

$$\text{minimize}_{y} \quad d'y + \mu(Sy)$$

(LS) s.t. $\mathcal{N}y = b$

$$y \text{ binary}$$

Hence, the Lagrangian multiplier problem can be written as:

$$L^* = \max_{\mu} \left\{ \min_{y} \left\{ d'y + \mu(Sy) : \mathcal{N}y = b, y \in Binary \right\} \right\}$$

By Theorem II.5, the Lagrangian subproblem (LS) yields intrinsically integer solutions for any choice of μ. Thus, the optimal objective values of (LS) and its LP relaxation are equal.

Furthermore, by Proposition II.9, the optimal value L^* of the Lagrangian multiplier problem is equal to the optimal objective function value of the LP relaxation of (NS). Thus, even if we find an optimal value for μ, we cannot improve on the LP lower bound. However, the Lagrangian relaxation method may still be preferred if the convergence is faster than solving the LP relaxation of the SPP by standard means. We investigate this issue next.

Observe that the Lagrangian subproblem (LS) can be formulated as a directed acyclic shortest path problem. Hence, (LS) can be solved very efficiently with special network algorithms (e.g., [Ahuja et al. 1993, pp. 107-108]).

By assumption, for any given vector μ we can easily compute $L(\mu)$, so what is needed is a way to find a good μ (i.e., one that gives a strong upper bound $L(\mu)$). This

28

can be accomplished with a general iterative technique called *subgradient optimization*.

The kth step of the subgradient method is the following: Fixing the vector μ^k, we

compute an optimal solution y^k to the Lagrangian subproblem

$$\min_y \left\{ d'y + \mu^k (Sy) : \mathcal{N}y = b, \ y \ Binary \right\}.$$

Now, for a specified step size θ_k, we let

$$\mu^{(k+1)} = \mu^k + \theta_k (Sy^k)$$

and go to the $(k+1^{st})$ step.

Poljak [1967] provides a convergent step length sequence, but in practice this

sequence is very slow to converge and heuristic sequences are used. One such sequence

(e.g., [Bertsimas and Orlin 1991], [Caprara et al. 1999]) for selecting the step length θ_k is

defined by:

$$\theta_k = \frac{\lambda_k [UB - L(\mu^k)]}{\|Sy^k\|^2}$$

where UB is an upper bound on the optimal objective function value of (*NS*), and λ_k is a

scalar chosen between 0 and 2. Parameter λ_k controls the step-size along the subgradient

direction (Sy^k).

The classical Held-Karp approach (e.g., [Held and Karp 1971]) halves parameter

λ_k if for p consecutive iterations no lower bound improvement occurs. Caprara et al.

[1999] implement the following alternate strategy: λ_0 is set to 0.1. For every $p = 20$

subgradient iterations, the best and worst lower bounds computed on the last p

operations are compared. If these two values differ by more than 1%, the current value of λ is halved. If, on the other hand, the two values are within 0.1% of each other, the current value of λ is multiplied by 1.5. This last decision is motivated by the observation that either the current μ is almost optimal, or the smaller lower bound difference is a result of an excessively small step-size. Caprara et al. [1999] claim that compared with classical Held-Karp, this new approach leads to faster convergence to near-optimal multipliers.

Caprara et al. [1999] terminate subgradient optimization as soon as they estimate that the procedure converges to a near-optimal Lagrangian vector. This convergence occurs when the lower bound improvement obtained in the last 300 subgradient iterations is smaller than 1.0, and, in percentage, below 0.1%.

Marsten and Shephardson [1980] form a Lagrangian relaxation on the reformulated problem (*NS*) and use subgradient optimization (incorporated with the Held-Karp approach) to maximize the Lagrangian multiplier problem. Marsten and Shepardson report their computational experience on the two-duty problem and suggest further investigation on three- or four-duty period problems.

We also implement the Lagrangian relaxation on the reformulated problem (*NS*). The Lagrangian multiplier problem is solved using subgradient optimization, and the improvements reported by Caprara et al. [1999] are incorporated. Our computational results are presented in Section III.D.

B. EXTERIOR PENALTY METHOD

This section presents an exterior penalty method implemented on the column split SPP reformulation to obtain a good lower bound that can be computed with less effort than an LP relaxation of (SPP).

Courant by Bazaraa et al. [1993] suggest the use of penalty methods to solve constrained problems. Subsequently, Camp [1955] and Pietrgykowski [1962] discuss this approach to solve nonlinear problems. The latter reference also gives a convergence proof. Fiacco and McCormick [e.g., 1968] solve practical problems.

Let (NS_{REL}) denote the LP relaxation of the reformulated problem (NS). By moving constraints $Sy = 0$ to the objective function with a penalty parameter, we obtain the following *penalty function*:

$$P(\alpha) = \min_y \{d'y + \alpha(Sy)'(Sy) : \mathcal{N}y = b, y \geq 0\}$$

For a fixed value of α, the optimization problem in the right-hand side of the first equality is called the *penalty subproblem (PS)*. By expressing $\alpha(Sy)'(Sy)$ algebraically,

$$\alpha(Sy)'(Sy) = \sum_{j \in \Gamma} \sum_{k=1}^{|K_j|-1} \alpha_j^k (y_j^k - y_j^{k+1})^2$$

we can see that (PS) is clearly a *convex non-separable quadratic* programming problem with linear network flow constraints.

As α goes to infinity, $\alpha(Sy)'(Sy)$ goes to zero and constraints $Sy = 0$ are satisfied. Hence, the penalty function $P(\alpha)$ converges to the optimal objective function value of (NS_{REL}). In theory, the solution to the penalty problem can be made arbitrarily

31

close to the LP relaxation of the original SPP by choosing sufficiently large α. However, in reality if we choose a very large α and attempt to solve the penalty problem, we may have computational difficulties from the ill-conditioning we have induced. With a large α, more emphasis is placed on feasibility, and most procedures solving the penalty problem will quickly progress toward a feasible point. Even though this point may be far from the optimum, premature termination could occur [Bazaraa et al. 1993].

As a result of the above difficulties associated with very large penalty parameters, most algorithms using penalty functions employ a sequence of increasing penalty parameters such as the approach we take:

Initialization Step: Let $\varepsilon > 0$ be a termination scalar. Choose an initial point y_1, a penalty parameter $\alpha_1 > 0$, and a scalar $\beta > 1$. Let $k = 1$, and proceed to the main step.

Main Step:

1. Starting from y_k, solve the following penalty subproblem:

$$\min_{y \geq 0} \{d'y + \alpha(Sy)'(Sy) : \mathcal{N}y = b\}$$

   Let y_{k+1} be an optimal solution and go to Step 2.

2. If $\alpha_k(Sy)'(Sy) < \varepsilon$, stop; otherwise, let $\alpha_{k+1} = \beta\alpha_k$, replace k by $k+1$, and go to Step 1.

The convex quadratic non-separable continuous problem can be solved in polynomial time. However, with the integrality requirement, the problem becomes *NP-Hard* [Hochbaum 1993].

(*PS*) can be solved using specialized nonlinear network algorithms (e.g., Dembo [1987], Hearn et al. [1987]). Our attempts to obtain a quadratic network solver from these and other sources have been unsuccessful. We implement our penalty method using the CPLEX 6.6 [ILOG 2000] quadratic programming solver. The CPLEX quadratic programming solver uses a *barrier* method (see, Bazaraa et al. [1993, Chapter 9]) that is not specially designed for solving quadratic network flow problems. Nevertheless, we still obtain satisfactory performance from the CPLEX solver.

C.   ROW REORDERING

By reordering the rows, we may reduce the number of consecutive ones segments in the columns of an SPP. Hence, the Lagrangian subproblem or the penalty subproblem will be smaller and this may improve solution efficiency.

The optimal reordering of the rows to minimize the number of segments of consecutive ones in an SPP is a combinatorial optimization problem. Enumerating all row permutations and choosing the one with the minimum number of segments of consecutive ones is optimal. However, there are $m!$ row permutations, and computing the number of segments for each of these $m!$ orderings may be more difficult than solving the original SPP.

A more elegant way to state the row-ordering problem is as follows:

For each pair of rows (i, j), we find the number of columns that have a +1 entry in one row, but not the other. Let c_{ij} denote such a number for rows i and j. For instance, for row vectors a_i' and a_j' :

33

$$a_i' = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$a_j' = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$c_{ij} = 3.$$

Next, we form an undirected graph $G_A = (N, E)$, and define a node $i \in N$ for each row a_i' of A. We also define an artificial starting node $s \in N$. We join all nodes $i \ne j \in N$ by an edge $(i, j) \in E$, and assign the arc length c_{ij}. Let $c_{sj} = 0$ for all $j \in N$. Note that G_A is a *complete graph* (i.e., every pair of nodes in G_A is connected by an edge).

Given G_A, we determine a *tour* W (i.e., a cycle that visits each node in the network exactly once) with the smallest possible value of the tour length, $\sum_{(i,j) \in W} c_{ij}$. The order of the rows associated with ordered nodes $i \in N \setminus \{s\}$ in tour W is the optimal ordering that minimizes the number of segments of ones in an SPP.

Given a complete graph $G_A = (N, E)$, determining a tour W with the smallest possible value of the tour length, $\sum_{(i,j) \in W} c_{ij}$, is known as the Traveling Salesman Problem (TSP). TSP is perhaps the most famous problem in all of network and combinatorial optimization. In a colloquial description of the problem, a salesman must visit each of n cities exactly once and then return to his starting point. The time taken to travel from city i to j is c_{ij}. Find the order in which the salesman should make his tour so as to finish as quickly as possible.

A collection of papers tracing the history and research on TSP can be found in Lawler et al. [1985]. TSP belongs to the class of *NP-Complete* problems.

Solving the row-reordering problem to optimality is computationally expensive. However, some of the polynomial-time TSP heuristics can be used to obtain near-optimal results quickly.

A simple greedy approach for TSP (and hence the row-ordering problem) is the *nearest neighbor* heuristic. We start from node s, and at each iteration we reach a node that does not close a cycle and minimizes the new path constructed. In particular, after k iterations, we have a path $\{s, i_1, ..., i_k\}$ consisting of distinct nodes, and the next iteration, we add an arc (i_k, i_{k+1}) that minimizes $c_{i_k i}$ over all arcs with $i \neq s, i_1, ..., i_k$. After m iterations, all nodes are included in the path, which is then converted to a tour by adding the final arc (i_m, s).

Given a tour, we may try to improve its length by using a method that changes the tour incrementally. A popular method for TSP with $c_{ij} = c_{ji}$, $\forall i, j$, is the *k-OPT* heuristic. The k-OPT heuristic creates a new tour by exchanging k arcs of the current tour with another k arcs that do not belong to the tour. The k arcs are chosen to optimize the length of the new tour with $O(m^k)$. The method stops when no improvement of the current tour is possible through a k-interchange. For comprehensive information on the k-OPT heuristic and other TSP heuristics that can be incorporated to the row-reordering problem, the reader is referred to Cook et al. [1998].

D. COMPUTATIONAL RESULTS FOR THE SPP LOWER BOUND ALGORITHMS IMPLEMENTED WITH THE COLUMN SPLITTING REFORMULATION

In this section, we investigate the Lagrangian relaxation and the exterior penalty method with the column split reformulation for various problems including:

(i)   an instance of the aircraft carrier problem,

(ii)   sample data from two-, three-, and four-duty period problems, and

(iii) a subset of real-world airline crew scheduling problems.

We also explore the impact of row reordering on the solution times and lower bounds obtained using the Lagrangian relaxation and the penalty method.

All sample problems are solved using the CPLEX 6.6 [ILOG 2000] optimization solver on a Pentium III 650Mhz personal computer with 192Mb RAM. The Lagrangian relaxation procedure is implemented using the Compaq Visual Fortran [1999] programming language. The Lagrangian subproblems are solved using the network simplex solver GNET [Bradley et al. 1975].

Table III.1 shows computational results obtained for the Lagrangian relaxation procedure and the penalty method implemented on the reformulated problem (NS).

The test data with the SPPNW prefix are a subset of real-world airline crew scheduling problems (also used by Hoffman and Padberg [1993]) obtained from the online OR-Library [2000] presented by J.E. Beasley. Our computational experience shows that the number of consecutive ones segments for an airline crew scheduling problem is proportional to problem size (i.e., the number of consecutive ones segments increases as the numbers of rows and columns increase). The airline crew scheduling

36

problems tested in this chapter are small in size so that we can obtain modest numbers of consecutive ones segments, and consequently, we can demonstrate the convergence speed of our algorithms, using real-world data.

The sample data named Carrier is an instance of the aircraft carrier scheduling problem presented in Chapter VI. The rest of the data are generated randomly to obtain two-, three-, and four-duty period scheduling problems. The starting time, the length of each duty period in an alternate schedule, and the corresponding cost coefficient are generated from a uniform distribution with associated parameters. For a k-duty period scheduling problem, each alternate schedule is generated to contain at most k segments of consecutive ones.

Table III.1 lists the solution times and the lower bound values for each example obtained using the penalty method and Lagrangian relaxation, as well as the simplex, dual simplex, and barrier methods.

Data	Cols	Rows	Segs	LPLB	IP	Simplex	Dual	Bar	Penalty	Lagrangian Relaxation				
SPPNW23	711	19	1,427	12,317	12,534	0.03	0.04	0.06	0.60	Iters	10	100	1,000	10,000
										Time	0.01	0.61	4.61	40.59
										LB	5,238	7,409	8,507	9,977
SPPNW26	771	23	1,685	6,796	6,796	0.04	0.03	0.06	0.52	Iters	10	100	1,000	10,000
										Time	0.01	0.71	6.70	52.00
										LB	3,799	5,326	5,531	6,098
SPPNW28	1,210	18	3,905	8,169	8,298	0.05	0.05	0.08	0.98	Iters	10	100	1,000	10,000
										Time	0.17	1.32	11.20	29.00
										LB	4,736	6,504	6,903	7,092
SPPNW31	2,662	26	11,470	7,980	8,038	0.14	0.10	0.40	2.29	Iters	10	100	1,000	10,000
										Time	0.55	4.72	40.21	391.00
										LB	2,917	3,851	4,122	4,827
Carrier	2,248	622	544	447	460	0.18	0.13	0.25	0.70	Iters	4	10	1,000	10,000
										Time	0.04	0.09	7.69	83.00
										LB	305	418	440	440
Two-Duty I	2,612	129	60	423	431	0.84	0.64	0.90	0.45	Iters	2	4	100	10,000
										Time	0.01	0.02	0.42	39.00
										LB	398	407	416.3	422.20
Two-Duty II	5,896	256	51	1,244	1,318	1.10	0.89	1.52	0.63	Iters	3	10	1,000	10,000
										Time	0.05	0.10	8.00	96.00
										LB	1,136	1,198	1,239	1,239
Three-Duty I	4,434	523	313	6,236	6,353	2.16	0.88	1.40	0.83	Iters	4	10	100	10,000
										Time	0.07	0.09	0.93	105.00
										LB	4,513	5,192	6,205	6,228
Three-Duty II	6,158	541	367	4,622	4,953	2.63	1.16	1.83	1.01	Iters	5	10	100	10,000
										Time	0.10	0.34	3.70	43.00
										LB	4,236	4,449	4,513	4,615
Four-Duty I	8,032	541	2,187	12,243	12,450	14.17	2.78	2.86	2.89	Iters	10	100	1,000	10,000
										Time	0.25	1.60	17.00	163.00
										LB	7,215	8,133	10,098	11,917
Four-Duty II	5,845	503	929	133	133	6.15	1.55	2.34	Not converged	Iters	2	3	100	10,000
										Time	0.04	0.04	1.23	122.00
										LB	119	132.65	132.65	132.65
Four-Duty III	5,845	541	1,758	4,155	4,432	29.16	7.27	3.15	2.68	Iters	2	5	100	10,000
										Time	0.08	0.10	1.47	150.00
										LB	3,533	3,823	3,823	3,852

Table III.1. Comparison of Lagrangian Relaxation and Exterior Penalty Methods with Linear Programming Solvers. From left to right, Cols, Rows, and Segs refer to the numbers of columns, rows, and consecutive ones segments, respectively. LPLB denotes the lower bound value obtained by solving the LP relaxation of the original SPP. IP is the optimal objective value of the original SPP. Simplex, Dual, and Bar refer to LP relaxation solution times using the simplex, dual simplex, and barrier methods, respectively. Iters denotes the number of subgradient iterations. LB is an abbreviation for lower bound. Time and lower bound results are listed for various numbers of subgradient iterations. The *Not converged* statement for the Four Duty II sample problem means that the penalty method does not converge to the optimal solution, due to ill-conditioning. All times in the table are in 650Mhz Pentium III seconds.

38

For airline crew scheduling sample problems that do not exhibit a consecutive ones structure, the convergence of Lagrangian relaxation is not satisfactory. By contrast, for most of the sample duty scheduling problems, Lagrangian relaxation yields a lower bound faster than the other methods. However, obtaining a lower bound within 1% of the LP relaxation lower bound using Lagrangian relaxation usually requires an excessive number of iterations and long solution times.

For all sample problems except Four-Duty II, the penalty method converges to the LP lower bound in at most three iterations. It fails to converge for Four-Duty II.

Our computational experience shows that, for certain problems with consecutive ones structure, the Penalty method presented yields lower bound values faster than the simplex, dual simplex, and barrier methods. Using a specialized quadratic network solver would presumably result in even further improvements in the penalty subproblem solution times. However, the reliability of this technique, especially on larger problems, is suspect.

Next, we investigate the impact of row reordering on the solution times and lower bounds obtained using the penalty method and the Lagrangian relaxation. Because the carrier model and the k-duty period problems in Table III.1 are generated with intrinsic row ordering that exhibit columns with consecutive ones, we exclude these examples from our experimentation. The airline crew scheduling sample problems that do not exhibit a consecutive ones structure are of special interest. Given the existing row order, a 2-OPT TSP heuristic is used to minimize the number of segments of consecutive ones for each sample problem.

Table III.2 lists the solution times for each sample problem using the penalty method before and after implementing a 2-OPT row-reordering heuristic. The sample problems are obtained from the online OR-Library [2000] presented by J.E. Beasley. Because the number of airline crew scheduling samples in Table III.1 is only four, we include additional samples of various sizes.

Data	Cols	Rows	OrgSegs	ReordSegs	%DecSegs	ReordTime	LPRel	OrgPen	ReordPen	%DecTime
SPPNW06	6,774	50	32,566	20,161	38	0.50	0.39	16.72	12.86	23
SPPNW07	5,172	36	20,360	13,954	31	0.22	0.23	6.65	4.44	33
SPPNW09	3,103	40	11,354	7,748	32	0.11	0.25	5.20	2.59	50
SPPNW11	8,820	39	33,065	24,786	25	0.38	0.39	21.12	11.65	45
SPPNW23	711	19	1,427	1,013	29	0.00	0.03	0.60	0.31	48
SPPNW26	771	23	1,685	1,222	27	0.00	0.04	0.52	0.33	37
SPPNW27	1,355	22	4,372	2,634	40	0.00	0.14	1.25	0.83	34
SPPNW28	1,210	18	3,905	2,450	37	0.00	0.05	0.92	0.60	35
SPPNW29	2,540	18	7,322	5,516	25	0.05	0.13	4.43	2.87	35
SPPNW31	2,662	26	11,470	7,112	38	0.05	0.14	4.59	2.21	52
SPPNW33	3,068	23	11,117	8,243	26	0.11	0.26	5.83	2.81	52
SPPNW35	1,709	23	4,839	3,887	20	0.06	0.18	2.26	1.12	50
SPPNW36	1,783	20	6,823	3,883	43	0.06	0.27	2.92	1.70	42
SPPNW38	1,220	23	3,975	2,258	43	0.00	0.16	1.72	0.73	58
SPPNW43	1,072	18	2,443	1,899	22	0.00	0.56	0.70	0.54	23

Table III.2. Comparison of the Solution Times Obtained Using the Exterior Penalty Method before and after Row Reordering. From left to right, Cols, Rows, OrgSegs, and ReordSegs refer to the numbers of columns, rows, and consecutive ones segments before and after row reordering, respectively. %DecSegs denotes the percent decrease in the number of consecutive ones segments after row reordering. Reordtime is the time to implement the 2-OPT row-reordering heuristic. LPRel denotes the LP relaxation solution time using the simplex method. OrgPen and ReordPen refer to the solution times using the exterior penalty method before and after row reordering, respectively. %DecTime denotes the percent decrease in the solution time after row reordering. All times in the table are in 650Mhz Pentium III seconds.

By reordering the rows of each sample problem using a 2-OPT TSP heuristic, we obtain a 32% average decrease in the number of segments of consecutive ones. Furthermore, the solution times obtained using the exterior penalty method before and

after reordering the rows improve an average of 41%. For SPPNW43, after reordering, we obtain a better solution time using the exterior penalty method than solving the original LP relaxation using the simplex method.

Table III.3 lists the solution times and the lower bound values obtained using the Lagrangian relaxation before and after reordering the rows for each example in Table III.2.

					Before Reordering				After Reordering			
Data	LPLB	IP	LPRel	Iters	10	100	1,000	10,000	10	100	1,000	10,000
SPPNW06	7,640.0	7,810	0.39	Time	2.09	15.16	128.00	1278	1.21	10.00	99.69	870.00
				LB	1,897	3,128	3,385	3,949	1,972	3,399	3,687	4,507
SPPNW07	5,476.0	5,476	0.23	Time	1.26	10.22	84.00	797	0.82	6.26	52.00	558.00
				LB	2,484	3,854	4,073	4,630	2,948	4,311	4,650	5,002
SPPNW09	67,760.0	67,760	0.25	Time	0.72	5.44	49.27	460	0.39	3.29	28.34	252.00
				LB	7,263	13,130	14,268	16,451	10,029	14,165	15,841	16,947
SPPNW11	11,6254.5	116,256	0.39	Time	2.20	14.89	117.00	1,184.00	2.03	13.96	113.00	1,080.00
				LB	9,252	13,375	14,422	17,936	10,155	14,717	16,336	21,631
SPPNW23	12,317.0	12,534	0.03	Time	0.01	0.61	4.61	40.59	0.00	0.44	4.22	33.00
				LB	5,238	7,409	8,507	9,977	5,959	7,715	8,922	10,062
SPPNW26	6,796.0	6,796	0.04	Time	0.01	0.71	6.70	52.00	0.00	0.44	3.73	31.00
				LB	3,799	5,326	5,531	6,098	4,725	5,526	5,933	6,456
SPPNW27	9,877.5	9,933	0.14	Time	0.22	1.59	12.30	38.72	0.11	0.94	3.46	3.46
				LB	4,303	7,200	7,996	8,562	6,274	8,270	8,562	8,562
SPPNW28	8,169.0	8,298	0.05	Time	0.17	1.32	11.20	29.00	0.05	0.77	1.00	1.00
				LB	4,736	6,501	6,903	7,092	6,624	6,954	7,092	7,092
SPPNW29	4,185.3	4,274	0.13	Time	0.33	3.13	27.79	284.00	0.28	2.04	17.74	178.00
				LB	2,441	2,834	2,943	3,201	2,596	2,859	3,076	3,262
SPPNW31	7,980.0	8,038	0.14	Time	0.55	4.72	40.21	391.00	0.27	2.58	24.22	230.00
				LB	2,917	3,851	4,122	4,827	3,253	4,284	4,564	5,829
SPPNW33	6,484.0	6,678	0.26	Time	0.72	5.27	42.62	423.37	0.49	4.17	33.00	367.00
				LB	2,574	3,687	3,896	4,286	2,700	3,875	4,132	4,836
SPPNW35	7,206.0	7,216	0.18	Time	0.22	1.81	14.34	127.00	0.17	1.26	10.77	52.00
				LB	4,488	5,727	6,158	7,138	5,140	5,940	6,537	7,141
SPPNW36	7,260.0	7,314	0.27	Time	0.27	2.63	24.55	209.00	0.17	1.42	12.46	100.00
				LB	2,568	4,137	4,434	5,061	3,217	4,505	4,819	5,318
SPPNW38	5,552.0	5,558	0.16	Time	0.16	1.42	10.71	125.00	0.11	0.77	4.94	5.00
				LB	3,540	4,370	4,561	4,798	4,072	4,727	4,798	4,798
SPPNW43	8,897.0	8,904	0.56	Time	0.11	0.88	7.08	69.00	0.06	0.66	5.55	53.00
				LB	5,513	7,283	7,847	8,438	6,013	7,617	8,125	8,438

Table III.3. Comparison of the Solution Times and Lower Bound Values Obtained Using the Lagrangian relaxation before and after Row Reordering. LPLB denotes the lower bound value obtained by solving the LP relaxation of the original SPP. IP is the optimal objective value of the original SPP. LPRel denotes the LP relaxation solution time using the simplex method. Iters denotes the number of subgradient iterations. LB is an abbreviation for lower bound. Solution times and lower bound values obtained using the Lagrangian relaxation before and after row-reordering are displayed for various numbers of subgradient iterations. All times in the table are in 650Mhz Pentium III seconds.

After reordering the rows, the solution times and the lower bound values obtained using the Lagrangian relaxation improve an average of 41% and 13%, respectively.

Our computational results show that, by using a row reordering heuristic, or by logically ordering the rows of the seminal SPP (e.g., Bausch et al. [1995, pg. 7]) to obtain minimal consecutive ones segments, the SPP lower bound algorithms presented here can be implemented with significantly less computational effort.

We also observe that the convergence rate of the Lagrangian relaxation and the exterior penalty methods are inversely proportional to the number of consecutive ones segments. That is, the efficiency decreases as the number of ones segments increases.

The airline crew scheduling problems tested here are small. However, the ratio of the number of segments to the number of columns is relatively high. Because the size of each sample problem is small, the LP relaxation of each sample SPP can be solved in a short amount of time.

On the other hand, the convergence rate of the exterior penalty method and the Lagrangian relaxation are more dependent to the number of consecutive ones segments than on the size of the seminal SPP. When we reorder the rows of each sample problem using a 2-OPT heuristic, the size of the sample does not change, but we reduce the number of consecutive ones segments. Moreover, when we reduce the number of consecutive ones segments for each sample problem, the exterior penalty method and the Lagrangian relaxation converge faster.

THIS PAGE INTENTIONALLY LEFT BLANK

IV.   INTEGRATING A ROW SPLIT TECHNIQUE WITH BENDERS DECOMPOSITION

This chapter presents a new algorithm to solve binary programming problems (e.g., SP, SC, or SPP) whose rows contain segments of consecutive ones. The algorithm first constructs a reformulation of the original binary problem by splitting each row to obtain exactly one segment of consecutive ones and additional auxiliary variables in the newly formed rows. The auxiliary variables are used to link the split rows so that they collectively represent equivalent feasible solutions. The special structure in the reformulated problem invites the use of Benders decomposition to solve, or at least bound the solution of SP, SC or SPP. The performance of the new algorithm is tested on a subset of the U.S. Navy aircraft carrier scheduling problem, and the airline crew scheduling data obtained from OR-Library [2000].

A.   ROW SPLITTING TECHNIQUE

Consider the SPP given by (*SPP*) in Chapter II.

In (*SPP*), for each row vector a_i' we define a *segment of ones* as a consecutive set of column elements a_{ij}, $j = l_i, l_i + 1, ..., l_i + p_i - 1, l_i + p_i$, such that $a_{ij} = 1$ for $j = l_i, ..., l_i + p_i$, where $a_{i,l_i-1} = 0$ if $l_i > 1$ and $a_{i,l_i+p_i+1} = 0$ if $l_i + p_i < n$. Let K_i be the set of segments of ones for row i containing the information where each segment starts and ends in terms of pairs of ordered column indices (e.g., $K_i = \{(l_i, l_i + p_i), ...\}$). $|K_i|$ denotes the number of segments in row i. Let Φ be the set of rows that have more than one segment of ones (i.e., $\Phi = \{i : |K_i| > 1\}$). Let a_i^k be a vector of size n, such that

$a_{ij}^k = 1$ if the jth element of row i is in the kth segment of K_i in (SPP), and $a_{ij}^k = 0$ otherwise. Note that the sum of a_i^k for all k is equal to the vector a_i (i.e., $\sum_{k=1}^{|K_i|} a_i^k = a_i$).

Using the split vectors a_i^k, (SPP) can be equivalently formulated as:

$$\text{minimize} \quad \sum_{j=1}^{n} c_j x_j \tag{IV.1.a}$$

$$(RS) \qquad \text{s.t.} \quad \sum_{j=1}^{n} a_{ij}^1 x_j = 1 \qquad \forall\, i \notin \Phi \tag{IV.1.b}$$

$$\sum_{j=1}^{n} a_{ij}^1 x_j + \sum_{k=1}^{|K_i|-1} s_i^k = 1 \quad \forall\, i \in \Phi \tag{IV.1.c}$$

$$\sum_{j=1}^{n} a_{ij}^k x_j - s_i^{k-1} = 0 \qquad \forall\, i \in \Phi, k = 2,\ldots,|K_i| \tag{IV.1.d}$$

$$x_j, s_i^k \text{ binary} \qquad \forall\, i \in \Phi,\, j \text{ and } k = 1,\ldots,|K_i|-1 \tag{IV.1.e}$$

Binary variables s_i^k are used to bind the split vectors a_i^k, $k = 1,\ldots,|K_i|$, to each other so that the original constraints of (SPP) are preserved.

Using matrix notation, (RS) can be written as:

$$\text{minimize} \quad c'x$$

$$(RS) \qquad \text{s.t.} \quad Fx + Gs = b$$

$$x, s \text{ binary}$$

46

Proposition IV.1: If (x^*, s^*) is an optimal solution to (RS), then x^* is an optimal

solution to (SPP).

Proof: The objective functions of (SPP) and (RS) are the same. We need to show that

the feasibility of each original constraint in (SPP) is preserved in the reformulated

problem (RS). If $|K_i| = 1$ for some row i of (SPP), then row i is unchanged in (RS).

However, if $|K_i| > 1$, then row vector a_i' is split into vectors a_i^k for all $k = 1, ..., |K_i|$.

Suppose that $|K_i| > 1$ for some row i in (SPP). Let N_i be the set of column indices that

have non-zero entries in row i. To satisfy constraint i, exactly one of the variables x_j

must be equal to one for all $j \in N_i$. Suppose, $x_l = 1$ for a feasible solution to (SPP). Let

the non-zero entry corresponding to x_l be in segment r of row i (i.e., $a_{il}^r = 1$).

If r is the first segment, then $s_i^k = 0$ in (RS) for all $k = 1, ..., |K_i| - 1$.

Consequently, variables $x_j = 0$ for all $j \in N_i \setminus l$. If $r > 1$, then $s_i^{r-1} = 1$ implying that

$s_i^k = 0$ for all $k = \{1, ..., |K_i| - 1\} \setminus r$, and $x_j = 0$ for all $j \in N_i \setminus l$.

Hence, we have shown that the reformulated problem forces $x \leq 1$. Now we need

to show $x \geq 1$ is forced as well. Suppose $x_j = 0$ for all $j \in N_i$. Then, to satisfy

constraint i of (IV.1.c), exactly one of s_i^k must be equal to one for all $k = 1, ..., |K_i| - 1$, say

$s_i^r = 1$. However, this result violates constraint i, $k = (r + 1)$ of (IV.1.d). Thus, $x \geq 1$ is

also forced in (RS). QED

47

Using a similar logic it can be shown that the LP relaxation of (*RS*) and (*SPP*) are also equivalent.

Because matrix F of (*RS*) has exactly one segment of consecutive ones in each row, then the transpose matrix F' has exactly one segment of consecutive ones in each column. Hence, F' is an interval matrix. Interval matrices are TU (Corollary 2.10, [Nemhauser and Wolsey 1988, pg. 544]). By Proposition II.4, a matrix is TU if and only if the transpose of this matrix is also TU. Thus, F is a TU matrix.

Matrix G of (*RS*) has exactly one +1, and one −1 in each column. Therefore, by Proposition II.6, G is also a TU matrix.

Observe that the dual of the LP relaxation of (*RS*) can be solved using the Lagrangian relaxation or the exterior penalty method to obtain lower bound values on (*RS*) or (*SPP*).

The following example illustrates row splitting of a numerical SPP.

Example IV.2: Consider the following SPP where $m = 6$, and $n = 8$.

$$\text{minimize} \quad c'x$$

$$(SPP) \qquad \text{s.t.} \qquad Ax = e$$

$$x \ \textit{binary}$$

$$d = \begin{bmatrix} 3 & 2 & 5 & 6 & 9 & 4 & 3 & 5 \end{bmatrix}$$

$$A = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{vmatrix}$$

48

We split the rows of A to obtain the following matrix, F.

$$F=\begin{array}{c|cccccccc}
a_1^1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
a_1^2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
a_2^1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_2^2 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
a_3^1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
a_3^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
a_4^1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
a_5^1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_5^2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
a_5^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
a_6^1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
a_6^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}$$

Thus, the equivalent reformulation is as follows:

$$\min \quad 3x_1 +2x_2 +5x_3 +6x_4 +9x_5 +4x_6 +3x_7 +5x_8$$

$$\begin{aligned}
\text{s.t.} \quad & x_1 +x_2 +x_3 && +s_1^1 && && = 1 \\
& x_7 +x_8 && -s_1^1 && && = 0 \\
& x_1 && +s_2^1 && && = 1 \\
& x_4 +x_5 +x_6 && -s_2^1 && && = 0 \\
& x_3 +x_4 && && +s_3^1 && = 1 \\
& x_8 && && -s_3^1 && = 0 \\
& x_2 +x_3 +x_4 +x_5 && && && = 1 \\
& x_1 && && && +s_5^1 +s_5^2 = 1 \\
& x_4 +x_5 && && && -s_5^1 = 0 \\
& x_8 && && && -s_5^2 = 0 \\
& x_1 +x_2 +x_3 +x_4 && && && +s_6^1 = 1 \\
& x_8 && && && -s_6^1 = 0 \\
\end{aligned}$$

$$x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8,\ s_1^1,\ s_2^1,\ s_3^1,\ s_5^1,\ s_5^2,\ s_6^1 \in \{0,1\}$$

For instance, if $x_1 = 1$, then s_5^1 and s_5^2 must equal zero. Consequently, x_4, x_5, and x_8 must equal zero as well. Hence, the original constraint,

$$x_1 + x_4 + x_5 + x_8 = 1$$

is satisfied. If $x_8 = 1$, then $s_5^2 = 1$. Consequently, $s_5^1 = x_1 = x_4 = x_5 = 0$, and the original constraint above is still satisfied. Observe that $x_1 = x_4 = x_5 = x_8 = 0$ is infeasible in the reformulated problem.

End of Example IV.1

Different equivalent reformulations of (*SPP*) can also be derived using the row splitting technique. For instance, Constraints (IV.1.c) and (IV.1.d) can also be written as:

$$\sum_{j=1}^{n} a_{ij}^1 x_j + s_i^1 = 1 \qquad \forall i \in \Phi \qquad \text{(IV.2.a)}$$

$$\sum_{j=1}^{n} a_{ij}^k x_j - s_i^{k-1} + s_i^k = 0 \qquad \forall i \in \Phi, k = 2, ..., |K_i| - 1 \qquad \text{(IV.2.b)}$$

$$\sum_{j=1}^{n} a_{ij}^{|K_i|} x_j - s_i^{|K_i|-1} = 0 \qquad \forall i \in \Phi \qquad \text{(IV.2.c)}$$

Note that the LP relaxation of the equivalent reformulation (IV.2) is equal to the LP relaxation of (*SPP*).

With minor adjustments, the arguments stated in this section for the reformulation of SPP also hold for SC and SP.

B. IMPLEMENTING BENDERS DECOMPOSITION ON THE ROW SPLIT REFORMULATION

The special structure in (RS) suggests the use of Benders decomposition.

Benders decomposition isolates and exploits special problem structure by fixing some of the variables. Benders decomposition for mixed-integer programming is described in Benders [1962]. For an expository review of Benders decomposition the reader is referred to Parker and Rardin [1988, pp. 237-249].

Consider the reformulated problem (RS).

$$\underset{x,s}{\text{minimize}} \quad c'x \qquad \qquad \text{(IV.3.a)}$$

(RS) s.t. $Fx + Gs = b$ (IV.3.b)

$$x, s \ \ binary \qquad \qquad \text{(IV.3.c)}$$

In Section A, we have shown that F is a TU matrix. To identify the special structure of F, we fix variable s in (RS) to a 0-1 vector $\hat{s}$, and move $G\hat{s}$ to the right-hand side of the equality (III.3.b). Thus, we obtain the following formulation:

$$\underset{x}{\text{minimize}} \quad c'x \qquad \qquad \text{(IV.4.a)}$$

(RSF) s.t. $Fx = b - G\hat{s}$ (IV.4.b)

$$x \ binary \qquad \qquad \text{(IV.4.c)}$$

The column vector $(b - G\hat{s})$ is integral. By Theorem II.5, the LP relaxation of (RSF), obtained by relaxing (IV.4.c) to $0 \le x \le 1$, yields integral solutions. Moreover, we can ignore the upper bound $x \le 1$ for an SPP, because x intrinsically cannot be greater than one. Thus, (RSF) can be equivalently written as:

$$\text{minimize} \quad c'x \qquad\qquad\qquad\qquad\qquad\qquad\text{(IV.5.a)}$$
$$\underset{x}{}$$

$$(RSF) \qquad \text{s.t.} \qquad Fx = b - G\hat{s} \qquad\qquad\qquad\qquad\qquad\text{(IV.5.b)}$$

$$x \geq 0 \qquad\qquad\qquad\qquad\qquad\qquad\text{(IV.5.c)}$$

Because variables s are fixed to $\hat{s}$, (RSF) is a restriction of (RS). Hence, the objective function value of (RSD) is an upper bound on the objective value of (RS).

Taking the *dual*[†] of (RSF), we obtain the following problem:

$$\text{maximize} \quad p'(b - G\hat{s}) \qquad\qquad\qquad\qquad\qquad\text{(IV.6.a)}$$
$$\underset{p}{}$$

$$(RSD) \qquad \text{s.t.} \qquad p'F \leq c \qquad\qquad\qquad\qquad\qquad\qquad\text{(IV.6.b)}$$

$$p \text{ free} \qquad\qquad\qquad\qquad\qquad\qquad\text{(IV.6.c)}$$

where the row vector p represents the dual variables. We may assume that c is an integral vector: by scaling c with a sufficiently large number, we can truncate c to an integral vector. This transformation does not appreciably affect the optimal value of x in (RSF). Thus, by Theorem II.5, if there exists a finite optimal solution to (RSD), then (RSD) will yield integral solutions.

Moreover, (RSD) can be transformed to a *minimum cost network flow problem*[‡]. Observe that the constraint coefficient matrix F' of (RSD) is an interval matrix. In Chapter II Section A, we show that an interval matrix can be transformed to a network matrix using elementary matrix operations.

[†]Duals and duality theory are introduced in introductory texts such as Bertsimas and Tsitsiklis [1997].

[‡]Network flow problems are introduced in introductory texts, e.g. Ahuja et al. [1993, Chapters 9, 10, 11]

52

Primal problem (*RSF*) is bounded; however it may be infeasible for some values of s. Without loss of generality, we may assume that there exists a feasible solution to (*RSF*) for some value of s. This assumption holds because (*RSF*) can be elasticized using artificial variables with sufficiently high cost coefficients in the objective function. If (*RSF*) is infeasible for some value of s, then (*RSD*) is either unbounded or infeasible by a corollary of the *duality theorem* of LP. Because p is a free variable and F is a 0-1 matrix, (*RSD*) cannot be infeasible. Thus, (*RSD*) can only be unbounded. We may avoid this problem either by elasticizing (*RSF*), or by forcing bounds on p in (*RSD*). Let us define some bounds on p in (*RSD*).

$$\text{maximize} \quad p'(b - G\hat{s}) \tag{IV.7.a}$$

$$(RSD) \quad \text{s.t.} \quad p'F \leq c \tag{IV.7.b}$$

$$l \leq p \leq u. \tag{IV.7.c}$$

Hence, by the *strong duality theorem* of LP, if (*RSF*) has an optimal solution, then so does (*RSD*), and the respective optimal costs are equal.

Let $\{p^1, p^2, ..., p^Q\}$ be the set of extreme points of $\{l \leq p \leq u : p'F \leq c\}$. Let $H < Q$. Then the problem,

$$\underset{z \text{ free}, \, s \in \{0,1\}}{\text{minimize}} \quad z \tag{IV.8.a}$$

$$(RSREL) \quad \text{s.t.} \quad z \geq (p^h)'(b - Gs) \qquad h = 1, ..., H \tag{IV.8.b}$$

is a relaxation of problem (*RS*). Problem (*RSREL*) is called *Benders relaxation* and problem (*RSF*) is called *Benders restriction*.

53

For fixed values of variable p, *Benders relaxation* problem yields a lower bound for the reformulated problem (*RS*). And, for fixed values of dual variable s, Benders restriction yields an upper bound. Benders decomposition algorithm solves Benders relaxation and Benders restriction problems alternately to obtain monotonically improving lower bounds (if the relaxation is solved exactly) and non-monotonic upper bounds, respectively. Because polyhedron $\{l \leq p \leq u : p'F \leq c\}$ has a finite number of extreme points, improved upper and lower bound values in Benders decomposition converge to an optimal solution in a finite number of iterations.

We present below an iteration diagram of the Benders decomposition algorithm implemented on the reformulated problem (*RS*).

Initialization Step: Select a convergence tolerance parameter $\varepsilon \geq 0$, and an integrality tolerance $\delta > 0$ (i.e., in the optimal solution to a B&B subproblem, the value of a variable is considered integral if it lies within δ of an integer) for the solution of Benders relaxation problems. Specify a termination scalar K that limits the number of iterations required. Let the initial upper bound (UB) be $+\infty$, and the initial lower bound (LB) be $-\infty$. Choose an initial point $\hat{s}$. Let $k = 1$, and proceed to the main step.

Main Step:

1. Solve the Benders restriction problem with the fixed values of s obtained either from the initialization step, or from the solution of the last Benders relaxation problem.

   Let $UB = \min\{UB,\ \text{optimal value of Benders' restriction}\}$.

2. If $k < K$ and $UB - LB > \varepsilon$, then

   Use optimal dual variables from the solution of the last Benders restriction problem to define a new constraint (*cut*) in the Benders relaxation problem.

   Solve Benders relaxation problem using $\delta > 0$.

   Let $LB = \max\{LB,\ \text{Optimal value of Benders' relaxation}\}$.

   Let $k = k + 1$

   Go to Step 1.

   Else,

   Stop.

   End of *if* statement.

The Bender's restriction problem (*RSD*) is a minimum-cost pure network flow problem. Hence, (*RSD*) can be solved very efficiently with special network flow algorithms (e.g., [Ahuja et al. 1993, Chapters 9, 10, 11]).

The number of columns in (*RSD*) is the same as the number of columns in the original (*SPP*). The number of rows in (*RSD*) is equal to the total number of segments of

ones in (*SPP*). The number of binary variables in the Benders relaxation problem (*RSREL*) is equal to the sum of segments of ones minus one, for each row in (*SPP*).

By reordering the columns, we may reduce the number of consecutive ones segments in the rows of an SP, SC, or SPP. Hence, the Bender's relaxation problem has smaller sizes that may further yield improved solution performance. The arguments stated in Chapter III, Section C for row reordering can similarly be used for column reordering, simply by replacing rows with columns.

C.   GENERATING IMPROVED FEASIBLE SOLUTIONS FOR SP AND SC PROBLEMS

1.   SP Problem

Using the same notation for (*SPP*), the row split reformulation problem for (*SP*) can be written as:

$$\underset{x}{\text{maximize}} \quad c'x \tag{IV.9.a}$$

$$(RSSP) \qquad \text{s.t.} \quad Fx + Gs \le b \tag{IV.9.b}$$

$$x, s \quad binary \tag{IV.9.c}$$

The Benders restriction and Benders relaxation problems for $(RSSP)$ are given by $(RESSP)$ and $(RELSP)$, respectively:

$$\underset{x}{\text{maximize}} \quad c'x \tag{IV.10.a}$$

$$(RESSP) \quad \text{s.t.} \quad Fx \le b - G\hat{s} \tag{IV.10.b}$$

$$x \ge 0 \tag{IV.10.c}$$

$$\underset{z \ \text{free}, \ s \in \{0,1\}}{\text{maximize}} \quad z \tag{IV.11.a}$$

$$(RELSP) \quad \text{s.t.} \quad z \le (p^h)'(b - Gs) \qquad h = 1,...,H \tag{IV.11.b}$$

$$\sum_{k=1}^{|K_i|-1} s_i^k \le 1 \qquad\qquad \forall i \in \Phi \tag{IV.11.c}$$

where p^h is the *dual price* vector obtained from $(RESSP)$ in iteration h of the Benders decomposition algorithm.

Constraints (IV.11.c) ensure that $(RESSP)$ is feasible for any value of $\hat{s}$ obtained from $(RELSP)$. Every feasible solution x to $(RESSP)$ is also feasible to (SP). Hence, by using the Benders decomposition algorithm, we can generate improved feasible solutions (i.e., feasible solutions having an improved objective value) for an SP problem.

2. SC Problem

Consider the following row split reformulation problem for (SC).

$$\text{minimize} \quad c'x \tag{IV.12.a}$$

$$(RSSC) \quad \text{s.t.} \quad Fx + Gs \ge b \tag{IV.12.b}$$

$$x, s \ \text{binary} \tag{IV.12.c}$$

The Benders restriction and Benders relaxation problems for $(RSSC)$ are given by $(RESSC)$ and $(RELSC)$, respectively:

$$\text{minimize} \quad c'x \tag{IV.13.a}$$

$$(RESSC) \quad \text{s.t.} \quad Fx \geq b - G\hat{s} \tag{IV.13.b}$$

$$x \geq 0 \tag{IV.13.c}$$

$$\underset{z \text{ free}, s \in \{0,1\}}{\text{maximize}} \quad z \tag{IV.14.a}$$

$$(RELSC) \quad \text{s.t.} \quad z \geq (p^h)'(b - Gs) \qquad h = 1,...,H \tag{IV.14.b}$$

Note that $(RESSC)$ is feasible for any value of $\hat{s}$ obtained from $(RELSC)$. Every feasible solution x to $(RESSC)$ is also feasible to (SC). Hence, the Benders decomposition algorithm can be used to generate improved feasible solutions for an SC problem.

$(RELSP)$ and $(RELSC)$ are integer programming problems which can be expensive to solve to optimality during the implementation of the Benders decomposition. By solving $(RELSP)$ and $(RELSC)$ to feasibility within a required *optimality tolerance* $\theta > 0$ (i.e., the objective value of the admissible feasible solution is within about $100.\theta$ % of the optimum), we may still obtain improved feasible solutions for an SP or SC problem, respectively (e.g., [Brown et al. 1987b]).

58

D. COMPUTATIONAL RESULTS FOR THE NEW ROW SPLIT BENDERS DECOMPOSITION (RSBD) ALGORITHM

The U.S. Navy aircraft carrier scheduling problem is one of many that inherently have long segments of consecutive ones in their rows. A partial list of other applications in the literature includes: Brownell and Lowerre [1976], Baker [1976], Bartholdi et al. [1980], Wing [1986], Brown et al. [1987a], Brown et al. [1990], Bausch et al. [1998]. Table IV.1 shows results with real data and the new row split Benders decomposition algorithm.

The sample problems are generated for twelve carriers and for various maintenance period *shifting* scenarios, using the ten-year schedule data taken from OPNAV Report 4710 [1996a]. Maintenance period shifting of an aircraft carrier is described in detail in Chapter VI.

Because it takes only one day for an aircraft carrier to transit from the Persian Gulf to the Mediterranean, sample problems are generated from a scenario that unifies the coverage of CENTCOM and EUCOM AORs. That is, the two AORs are considered as one, and the coverage is maximized (or the uncovered periods are minimized) for the unified AOR. We adapt this scenario to obtain various numbers of consecutive ones segments for each sample problem.

Table IV.1 shows the solution times for both the original SPP model and the new algorithm. Additionally, the number of Benders iterations is provided for each sample problem to indicate the convergence rate of the decomposition.

Problem	Cols	Rows	Segs	Ratio: Segs/Cols	Iters	CPLEX B&B	RSBD
1	877	579	30	0.03	2	1.1	0.1
2	896	581	43	0.05	4	1.2	0.2
3	914	582	60	0.07	1	1.2	0.1
4	932	609	93	0.10	4	1.4	0.3
5	945	621	114	0.12	4	1.7	0.3
6	925	629	155	0.17	3	1.5	0.2
7	954	638	162	0.17	5	1.7	0.3
8	960	643	173	0.18	5	1.7	0.3
9	972	643	189	0.19	4	1.6	0.4
10	1,024	667	230	0.22	5	1.9	0.4
11	1,113	676	244	0.21	5	1.9	1.7
12	1,168	682	431	0.37	17	2.3	3.4

Table IV.1. Computational Results for the Row Split Benders Decomposition. From left to right, Cols, Rows, and Segs refer to the numbers of columns, rows, and consecutive ones segments, respectively. Iters denotes the number of Benders iterations. Solution times for both the original SPP model and the row split Benders decomposition (RSBD) are listed as well as the number of Benders iterations (or cuts) generated by the row split Benders decomposition. Each original SPP is solved directly using CPLEX 6.6 [ILOG 2000]. All computations are implemented on a Pentium III 550Mhz personal computer with 384Mb RAM.

The new algorithm yields faster solution times for all sample problems but sample 12. Considering the numbers of Benders iterations, the convergence speed of the new algorithm is satisfactory for all sample problems.

Table IV.1 shows some instances where the row split Benders decomposition solves the set partition faster than conventional means. We conjecture that, given the difficulty of solving set partitions, there exist instances that the row split Benders decomposition will solve that defy conventional means.

Next, we investigate the impact of column reordering on the performance of our new algorithm. We also test the performance of the new algorithm in generating improved feasible solutions for some instances of SP, SC, or SPP.

Because the carrier model is generated with intrinsic column ordering that exhibits rows with consecutive ones, we omit it from our experimentation. Here, airline crew scheduling problems that do not exhibit a consecutive ones structure are of more interest. Our computational experience shows that the number of consecutive ones segments in the rows of an airline crew scheduling problem is proportional to the problem size (i.e., the number of consecutive ones segments increases as the numbers of rows and columns increase). The airline crew scheduling problems tested in this chapter are small so that we can obtain reasonable numbers of consecutive ones segments. Consequently, we can isolate the structure we seek using real-world data, and actually solve the SPPs with the row split Benders decomposition. The samples in Table IV.2 are even smaller than those in Tables III.2 and III.3, where we merely seek a good lower bound that can be computed with less effort than an LP relaxation of (*SPP*).

Here, we either attempt to solve the original integer programming problem to optimality or to obtain a feasible solution for it, using our new algorithm. In either case, the problems we try to solve are NP-Complete. Our computational experience shows that the convergence speed of the new algorithm decreases as the number of consecutive ones segments increases. Hence, to obtain modest numbers of consecutive ones segments, and accordingly to allow the size of the binary Benders relaxation problem to be computationally tractable, we choose smaller size airline crew scheduling problems in this chapter than the samples tested in Chapter III.

Given the existing column order, a 2-OPT TSP heuristic is used to minimize the number of segments of consecutive ones for each sample problem. Table IV.2 lists the

number of consecutive ones segments before and after column reordering, as well as the time to implement the 2-OPT heuristic.

Data	Cols	Rows	OrgSegs	ReordSegs	Ratio: OrgSegs/Cols	Ratio: ReordSegs/Cols	%DecSegs	ReordTime
SPPNW08	435	24	493	258	1.13	0.59	48	2.04
SPPNW23	711	19	552	272	0.78	0.38	51	9.66
SPPNW26	771	23	611	359	0.79	0.47	41	9.33
SPPNW28	1,210	18	971	538	0.80	0.44	45	42.00
SPPNW40	405	19	524	248	1.29	0.61	53	2.42
SPPNW41	198	17	229	122	1.16	0.62	47	0.22

Table IV.2. Comparison of the Number of Consecutive ones segments before and after Column Reordering. From left to right, Cols, Rows, OrgSegs, and ReordSegs refer to the numbers of columns, rows, and consecutive ones segments before and after column reordering, respectively. %DecSegs denotes the percent decrease in the number of consecutive ones segments after column reordering. Reordtime is the time to implement the 2-OPT column-reordering heuristic. All times in the table are in 650Mhz Pentium III seconds. The column reordering is implemented using Compaq Visual Fortran [1999].

By reordering the columns of each sample problem using a 2-OPT TSP heuristic, we obtain a 48% average decrease in the number of segments of consecutive ones.

Next, we investigate the performance of the new algorithm in generating feasible solutions for an SPP. Table IV.3 lists the solution times and the upper bound values obtained using the row split Benders decomposition algorithm before and after reordering the columns for each example in Table IV.2.

Data	IPObj	IPTime	θ	δ	Benders Decomposition							
SPPNW08	35,894	1.21	0.5	0.25	Iters	1	25	50	75	89	96	150
					TimeBe	0.3	78.4	189.5	372.6	471.7	528.0	1,112.0
					TimeAf	0.2	44.5	103.7	187.5	267.8	310.4	
					UBBe	12,276,342	5,337,328	3,657,892	2,345,726	934,587	934,587	899,452
					UBAf	7,227,440	3,623,764	1,345,687	957,484	75,394	38,613	
SPPNW23	12,534	0.71	0.6	0.30	Iters	1	25	50	75	100	111	150
					TimeBe	0.4	90.2	183.5	412.4	603.5	720.4	1,321.0
					TimeAf	0.3	49.9	123.3	201.1	324.2	342.6	
					UBBe	10,331,450	7,455,872	5,385,912	3,713,900	3,333,562	3,087,375	1,004,562
					UBAf	9,173,058	6,386,058	3,767,935	1,528,487	856,840	14,583	
SPPNW26	6,796	0.44	0.6	0.35	Iters	1	25	50	75	100	125	150
					TimeBe	0.2	100.3	199.4	476.5	631.3	1,087.0	1,434.0
					TimeAf	0.1	76.4	143.5	289.4	403.1	634.7	923.6
					UBBe	11,325,755	6,325,207	3,655,376	3,212,098	3,080,354	1,354,730	999,423
					UBAf	8,436,754	6,234,967	2,561,376	2,222,435	1,452,789	934,562	843,295
SPPNW28	8,298	0.51	0.8	0.40	Iters	1	25	50	75	100	125	150
					TimeBe	0.5	145.6	254.9	612.67	739.7	1,345	1,860
					TimeAf	0.4	103.3	194	439.5	625.7	738	1,406
					UBBe	7,455,127	6,513,538	6,004,577	4,987,002	3,056,276	2,377,553	1,235,740
					UBAf	8,345,265	6,345,850	5,349,012	4,333,587	3,265,290	1,458,873	634,547
SPPNW40	10,809	0.30	0.5	0.25	Iters	1	25	50	75	84	92	150
					TimeBe	0.3	87.4	198.3	390.6	493.2	538.1	1,243
					TimeAf	0.1	39.5	94.6	169.5	241.3	294.1	
					UBBe	10,343,566	6,341,846	4,234,745	3,100,328	1,349,162	1,000,237	763,310
					UBAf	5,435,767	2,546,854	1,875,045	878,098	53,234	14,207	
SPPNW41	11,307	0.25	0.4	0.10	Iters	1	23	26	28	30	34	84
					TimeBe	0.03	36.46	41.925	45.912	48.9	56.7	206.3
					TimeAf	0.006	6.6	7.65	8.41	9.21	10.89	
					UBBe	8,107,710	2,715,820	1,821,430	1,816,550	1,816,550	1,816,550	12,990
					UBAf	4,515,555	30,039	26,235	16,803	12,870	11,307	

Table IV.3. Comparison of the Solution Times and Upper Bound Values Obtained Using the Row split Benders Decomposition before and after Column Reordering for each SPP Sample. IPObj is the optimal objective value of the original SP problem. IPTime is the time to solve the original SP problem. θ and δ refer to the optimality and integrality tolerance parameters for Benders relaxation (*RSREL*), respectively. Iters denotes the number of Benders iterations. TimeBe and TimeAf refer to the times before and after column ordering, respectively. UBBe and UBAf refer to the upper bound values obtained using the new algorithm before and after column reordering, respectively. All times in the table are in 650Mhz Pentium III seconds.

63

Observe that when the number of consecutive ones segments is more than about 300, the row split Benders decomposition converges slowly for the airline crew scheduling samples in Table IV.2. Experiments with other airline crew scheduling problems from OR-Library [2000] behave similarly, and are not reported here.

As the number of cuts increases the time to solve the binary Benders relaxation increases significantly. For each of the airline crew scheduling samples, in order to obtain feasible solutions faster, we apply some heuristics (e.g., (i) during the initial iterations solve the LP relaxation of ($RSREL$), (ii) keep no more than k cuts in ($RSREL$) by eliminating the cuts having zero dual price for some number of iterations, (iii) increase the optimality tolerance θ, or the integrality tolerance δ).

After reordering the columns, the solution times and the upper bound values obtained using the new algorithm improve an average of 63% and 61%, respectively.

Next, we relax each SPP listed in Table IV.2 to obtain an SC sample problem. By using each sample problem, we investigate the performance of the row split Benders decomposition algorithm in generating improved feasible solutions for an SC problem. Table IV.4 lists the solution times and the upper bound values obtained using the new algorithm before and after reordering the columns for each example in Table IV.2.

Data	IPObj	IPTime	θ	Benders Decomposition									
SPPNW08	7,758	0.11	0.2	Iters	1	2	3	5	10	15	20	35	38	83
				TimeBe	0.01	0.05	0.10	0.21	0.76	1.43	2.30	10.85	12.79	76.23
				TimeAf	0.01	0.05	0.07	0.27	0.63	1.11	1.79	4.59	5.24	
				UBBe	275,346	275,346	193,528	62,038	20,220	16,354	16,354	14,336	13,480	8,318
				UBAf	248,150	60,908	51,440	25,120	19,660	13,462	10,552	9,458	8,154	
SPPNW23	10,062	0.15	0.2	Iters	1	2	3	5	10	16	18	45	47	86
				TimeBe	0.02	0.06	0.10	0.21	1.72	5.42	5.87	48.10	61.78	318.30
				TimeAf	0.02	0.06	0.09	0.16	1.05	2.55	2.79	11.66	12.10	
				UBBe	549,614	529,712	183,504	118,916	56,034	46,682	36,696	23,300	23,198	11,088
				UBAf	228,732	168,430	72,002	38,236	17,032	15,644	13,538	11,810	11,298	
SPPNW26	6,458	0.20	0.3	Iters	1	2	3	5	10	16	27	56	73	93
				TimeBe	0.02	0.07	0.12	0.30	1.64	6.03	16.48	52.53	83.67	137.00
				TimeAf	0.03	0.06	0.10	0.18	0.86	2.12	4.54	18.99	38.07	
				UBBe	759,040	361,338	141,336	33,824	33,824	29,012	27,054	18,686	18,028	7,560
				UBAf	285,438	118,244	54,732	24,252	18,974	16,130	10,486	9,300	7,954	
SPPNW28	7,092	0.70	0.3	Iters	1	2	3	5	10	25	65	80	87	113
				TimeBe	0.02	0.07	0.18	0.38	1.12	23.25	313.80	518.40	580.74	1,196.00
				TimeAf	0.02	0.07	0.12	0.25	0.92	5.25	46.47	83.78	106.62	
				UBBe	349,506	349,506	349,506	349,506	55,215	27,366	25,341	12,477	12,477	7,239
				UBAf	551,751	347,844	303,552	74,256	25,005	13,956	12,240	9,744	8,892	
SPPNW40	10,701	0.16	0.3	Iters	1	2	5	10	15	20	30	35	40	94
				TimeBe	0.02	0.06	0.20	3.38	12.02	16.55	22.26	25.42	28.31	133.40
				TimeAf	0.01	0.01	0.13	1.74	2.16	2.75	4.89	6.33	7.77	
				UBBe	758,211	438,939	107,748	39,582	37,044	31,815	29,529	29,529	26,853	13,275
				UBAf	357,525	119,706	63,558	28,356	18,723	18,723	18,723	15,827	13,116	
SPPNW41	10,539	0.12	0.1	Iters	1	2	5	7	10	13	15	18	19	39
				TimeBe	0.03	0.01	0.11	0.52	2.68	7.85	9.32	23.84	30.75	78.54
				TimeAf	0.00	0.02	0.16	0.35	0.73	1.20	2.26	3.04	4.36	
				UBBe	400,464	184,668	51,672	42,528	31,614	19,035	26,178	19,035	19,035	10,539
				UBAf	177,720	56,592	37,728	30,903	25,050	15,816	15,816	15,816	10,539	

Table IV.4. Comparison of the Solution Times and Improved Feasible Solutions Obtained Using the Row split Benders Decomposition before and after Column Reordering for each SC Sample. IPObj is the optimal objective value of the original SC problem. IPTime is the time to solve the original SC problem. θ refers to the optimality tolerance for Benders relaxation, (*RELSC*). The integrality tolerance δ is set to the CPLEX 6.6 [ILOG 2000] default value (1.0e–5). Iters denotes the number of Benders iterations. TimeBe and TimeAf refer to the times before and after column ordering, respectively. UBBe and UBAf refer to the upper bound values obtained using the new algorithm before and after column reordering, respectively. All times in the table are in 650Mhz Pentium III seconds

By reordering the columns of each sample problem, the number of Benders iterations required to obtain a feasible solution within a required optimality tolerance θ decreases an average of 42%. Furthermore, the solution times improve an average of 90%.

We also use the SPPs listed in Table IV.2 to obtain SP sample problems. By using each sample problem, we investigate the performance of the row split Benders decomposition algorithm in generating improved feasible solutions for an SP problem. Table IV.5 lists the solution times and the lower bound values obtained using the new algorithm before and after reordering the columns for each SP example.

Data	IPObj	IPTime	θ	Benders Decomposition							
				Iters	1	2	3	4	5	6	8
				TimeBe	0.03	0.04	0.08	0.11	0.14	0.16	
SPPNW08	174,793	0.26	0.05	TimeAf	0.05	0.07	0.09	0.12	0.15	0.17	0.24
				LBBe	28,426	28,426	28,426	28,426	28,426	173,872	
				LBAf	34,614	34,614	34,614	34,614	34,614	34,614	173,330
				Iters	1	24	35	40	50	60	94
				TimeBe	0.03	5.86	15.59	17.59	25.33	32.91	
SPPNW23	28,644	0.67	0.20	TimeAf	0.03	2.95	10.61	14.26	21.86	58.15	199.70
				LBBe	4,778	19,080	22,628	24,256	24,256	28,360	
				LBAf	10,158	13,338	16,052	18,200	19,190	21,186	28,464
				Iters	1	6	40	45	62	83	114
				TimeBe	0.04	0.31	11.01	15.49	25.65	45.32	121.54
SPPNW26	28,560	0.31	0.20	TimeAf	0.05	0.27	6.59	7.69	14.61	37.22	
				LBBe	3,926	3,926	10,578	10,578	18,792	22,572	23,124
				LBAf	9,162	11,138	14,890	20,120	24,006	24,214	
				Iters	1	33	35	39	45	56	85
				TimeBe	0.03	14.66	17.38	20.69	23.35	32.23	73.15
SPPNW28	30,744	0.45	0.30	TimeAf	0.08	5.11	5.53	6.78	8.97	12.37	
				LBBe	7,593	12,351	12,351	17,589	17,589	20,169	28,122
				LBAf	11,403	20,589	20,820	22,092	24,018	28,110	
				Iters	1	9	23	26	29	32	73
				TimeBe	0.01	0.46	15.67	20.36	20.10	21.66	42.90
SPPNW40	34,137	0.17	0.10	TimeAf	0.01	0.41	2.16	2.87	3.21	4.77	
				LBBe	6,900	11,298	11,298	16,035	16,035	16,287	32,064
				LBAf	5,265	15,462	25,890	25,890	25,890	32,679	
				Iters	1	9	23	26	29	32	73
				TimeBe	0.01	0.46	15.67	20.36	20.99	21.66	42.90
SPPNW41	37,845	0.10	0.05	TimeAf	0.01	0.41	2.16	2.87	3.21	4.77	
				LBBe	6,900	11,298	11,298	16,035	16,035	16,287	32,064
				LBAf	5,265	15,462	· 25,890	25,890	25,890	32,679	

Table IV.5. Comparison of the Solution Times and Improved Feasible Solutions Obtained Using the Row split Benders Decomposition before and after Column Reordering for each SP Sample. IPObj is the optimal objective value of the original SP problem. IPTime is the time to solve the original SP problem. θ refers to the optimality tolerance for Benders relaxation, (*RELSP*). The integrality tolerance δ is set to the CPLEX 6.6 [ILOG 2000] default value (1.0e–5). Iters denotes the number of Benders iterations. TimeBe and TimeAf refer to the times before and after column ordering, respectively. LBBe and LBAf refer to the lower bound values obtained using the new algorithm before and after column reordering, respectively. All times in the table are in 650Mhz Pentium III seconds.

67

After column reordering, the numbers of Benders iterations and the times required to obtain feasible solutions within an optimality gap θ improve an average of 43% and 82%, respectively, for SPPNW26, SPPNW28, SPPNW40, and SPPNW41. However, for SPPNW08 and SPPNW23, the numbers of iterations and the solution times increase an average of 31% and 58%, respectively.

For SPPNW08, the row split Benders decomposition raises the lower bound (hence, generating a feasible solution) within 5% of the optimal solution faster than solving the seminal SP problem using CPLEX 6.6 [ILOG 1999].

In general, our computational results show that the convergence rate of the new algorithm is inversely proportional to the number of consecutive ones segments. That is, the efficiency decreases as the number of ones segments increases.

The airline crew scheduling problems tested here are small. However, the ratio of the number of segments to the number of columns is relatively high. Because the size of each sample problem is small, the seminal SP, SC, or SPP can be solved in a short amount of time.

On the other hand, the size of the binary Benders relaxation problem, and hence the convergence rate of the new algorithm is more dependent to the number of consecutive ones segments than on the size of the original SP, SC, or SPP. When we reorder the columns of each sample problem using a 2-OPT heuristic, the size of the sample does not change, but we reduce the number of consecutive ones segments. Moreover, in general, when we reduce the number of consecutive ones segments for each sample problem, the new algorithm converges faster.

68

We conjecture that a large SC, SP or SPP with a modest number of consecutive ones segments can be solved faster with the row split Benders decomposition than by conventional means.

The row splitting technique is implemented using Compaq Visual Fortran [1999], and Benders decomposition is implemented using the AMPL Plus 1.6 [Fourer et al. 1999] mathematical programming language and the CPLEX 6.6 [ILOG 2000].

Execution of an indirect solution method such as Benders decomposition is not nearly as fast or efficient in a mathematical modeling language such as AMPL as it is in a specialized, purpose-built code (e.g., [Brown et al. 1987b]). That is, solution times obtained using the new algorithm could be improved by implementing Benders decomposition using a specialized code.

THIS PAGE INTENTIONALLY LEFT BLANK

V.   REDUCTIONS IN SPP

Problem size reduction is essential for efficient solution of SPPs. Preprocessing finds "embedded special structure that can give significant insight to the model proponent as well as greatly reduce solution effort." [Brown et al. 1980]

In this chapter, we first present reduction operations established in the literature for SPPs. Then, we introduce other reductions suggested by the network with side constraints reformulation of SPP, using the column splitting technique. Next, we show the relationship between these network reductions and the known SPP reductions. While doing this, we discover a new reduction in the column split reformulation than has an equivalent in SPP that is new to the literature. Applying this *column split reduction* to a well-studied set of airline crew scheduling SPPs, we identify relations between binary variables that lead to fixing many more of them than any preceding work. We also discuss a variable *probing* method (i.e., tentatively setting the variable to one of its bounds and observing the implications of this setting) for SPP that can be used to tighten the LP relaxation lower bound, or reduce the problem size during B&B.

A.   KNOWN REDUCTION OPERATIONS

We will begin with technical definitions used to describe reduction operations. Consider the integer programming formulation (*SPP*) introduced in Chapter I. *Fixing a variable x_j to zero* means that the variable, its objective function coefficient c_j, and the corresponding column j are permanently removed from the problem formulation. *Removing a row i* means eliminating that row from the problem matrix along with the

corresponding right-hand side entry, and fixing any variable to zero whose resulting column now has only zero entries. *Fixing a variable x_j to one* means that all its incident rows are satisfied and can be removed. Moreover, all other columns incident with any of these rows can be fixed to zero.

Sometimes columns are *merged* during reduction, which means that the (orthogonal) columns of two variables are combined into one column, and the two original columns are deleted from the formulation. The objective function coefficient of the merged variable is the sum of the two original objective function coefficients.

In the following subsections we describe the published reduction methods (e.g., [Balas and Padberg 1976], and [Hoffman and Padberg 1993]).

1.   Duplicate Columns

If two columns are identical, then the column with the larger cost coefficient can be removed from the problem.

- $A_j = A_k$ for some $j, k \in N$ such that $j \neq k \Rightarrow$

    If $c_j > c_k$, then x_j is fixed to zero, else x_k is fixed to zero.

Duplicate columns do arise especially in large-scale SPPs. For instance, in the airline crew scheduling problem, because the number of feasible rotations is astronomically large, a very large number of pairings is generated according to an appropriate heuristic sieve. Because the process of generating such problems is both random and heuristic, the same column may be generated more than once.

72

2. A Column Is Equal To The Sum Of Other Columns

If the sum of a subset of columns in the formulation is equal to a single column, and the total cost of the columns in the sum is smaller than the cost of the single column, then this column can be removed from the problem.

- $A_j = \sum_{k \in K} A_k$ and $c_j \geq \sum_{k \in K} c_k$ for some $j \in N$ and $K \subseteq N \setminus \{j\} \Rightarrow$

  x_j is fixed to 0.

3. Dominated Rows

If a row in (SPP) properly contains another row, then the longer row (i.e., the row with more non-zero entries) can be removed from the problem along with the variables that have non-zero coefficients in the longer row but not in the shorter row. Let N_i ($\subseteq N$) be the set of column indices that have non-zero entries in row i (i.e., $N_i = \{ j \in N \mid a_{ij} = 1 \}$).

- $N_i \subseteq N_l$ for some $i, l \in M$ and $i \neq l \Rightarrow$

  x_j is fixed to 0 $\forall j \in (N_l \setminus N_i)$, row l is removed.

4. Two Rows Differ By Two Entries

If two rows are identical except for two entries, then the two columns corresponding to these entries can either be removed from the formulation (if they are non-orthogonal) or merged into one column (if they are orthogonal). In either case, the two rows will be identical after the variable reductions. Thus, one of the rows can be removed from the formulation.

73

- $|N_i| = |N_l|$, $N_i \setminus (N_i \cap N_l) = \{j\}$, and $N_l \setminus (N_i \cap N_l) = \{k\}$ for some $i, l \in M \Rightarrow$

  If $A_j' A_k \geq 1$ then x_j, x_k are both fixed to zero, else x_j and x_k are merged. One

  of the rows is removed in each case.

5.   Singleton Row

If a row has exactly one non-zero entry, then the variable corresponding to this

entry can be fixed to one.

- $a_{ij} = 1$ for $j \in N$, and $a_{ik} = 0$ $\forall k \in N \setminus \{j\}$ for some $i \in M \Rightarrow$

  x_j is fixed to 1.

6.   Clique Reduction

Let $G_A = (N, \mathbb{E})$ be an *intersection graph* associated with the incidence matrix A.

The intersection graph G_A is obtained as follows.

We define a node $j \in N = \{1, 2, ..., n\}$ for each column A_j of A, and join two

nodes $i \neq j \in N$ by an edge $(i, j) \in \mathbb{E}$ if the columns A_i and A_j of A have at least one

entry equal to $+1$ in common in some row of A (i.e., if A_i and A_j are non-orthogonal).

Example V.1 illustrates the intersection graph for a 0-1 matrix, A.

Example V.1: Consider the following 0-1 matrix.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure V.1 depicts the intersection graph of matrix A.
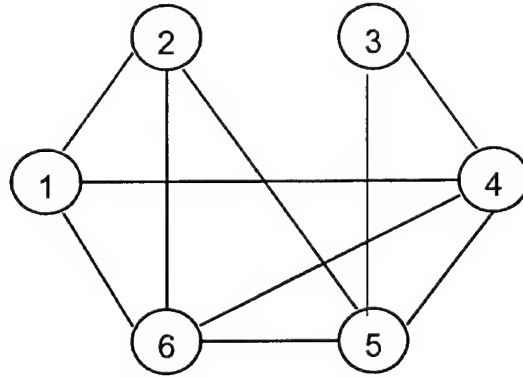
Figure V.1   Intersection Graph G_A for Sample Matrix A.   The nodes of G_A correspond to the columns of matrix A.   There exists an arc between two nodes if the corresponding column vectors are non-orthogonal.

For every feasible solution x of (*SPP*) there corresponds a disjoint node set $S = \{j \in N \mid x_j = 1\}$ in G_A. No two nodes of S are joined by an edge of G_A, otherwise x is not a feasible solution.

A *clique* is a set of nodes with the property that every pair of nodes in the set is connected by an edge.   Then, if C is a clique in the intersection graph G_A, $\sum_{j \in C} x_j \leq 1$ is a

valid inequality for (*SPP*).   That is, this inequality constraint is satisfied by all integer solutions to (*SPP*).   For instance, in Example V.1, nodes 1, 2, and 6 form a clique and

$$x_4 + x_5 + x_6 \leq 1$$

is a valid inequality for the associated SPP.

If a column with no intersection in row i is non-orthogonal to all the columns that have a non-zero entry in row i, then the variable corresponding to this column can be fixed to zero.   In terms of the intersection graph, a node extending a *row clique* (i.e., the

clique formed by the nodes associated with the non-zero entries in a row of SPP) can be removed.

- Clique Reduction:

    Let $k \in N$ be any column. $A_j \cdot A_k \geq 1 \ \forall \ j \in N_i$, for some $i \in M \Rightarrow$

    x_k is fixed to zero.

Finding a clique of a given size in some graph G is an NP-complete problem (e.g., [Parker and Rardin 1988, pg. 35]). However, given the definition above, finding clique reductions in an SPP is polynomial in the input size of the SPP. Given row i in SPP and the corresponding node set N_i, it is not difficult to determine the existence of a clique C that properly contains N_i. One simply scans all columns of A and distinguishes those that are non-orthogonal to all columns in N_i. Either this set is empty and a next row is selected, or variables that can be fixed to zero are detected.

The worst-case complexity of finding clique reductions in an SPP is $O(m \times n \times l)$, where m is the number of constraints, n is the number of columns, and l is the number of non-zero entries in matrix A. For each non-zero entry j in row i (i.e., $j \in N_i$), we scan all columns to distinguish those that are non-orthogonal to column j. In the worst-case, this operation is done for each row i, adding up to $(n \times l)$ comparisons. The worst-case complexity of comparing two columns to detect whether they are non-orthogonal is $O(m)$. Thus, the overall worst-case complexity is $O(m \times n \times l)$.

Clique reduction is used as a part of the column split reduction that is introduced in Section V.D.

76

Figure V.2 illustrates the reductions described in this section.

(c) Duplicate columns j and k

(b) Column j is equal to and dominated by the sum of other columns

(c) Row i dominates row l, and columns $j \in J$ are fixed to 0

(d) Rows i and l differ by two entries in columns j and k

(e) Singleton Row

(f) Clique Reduction
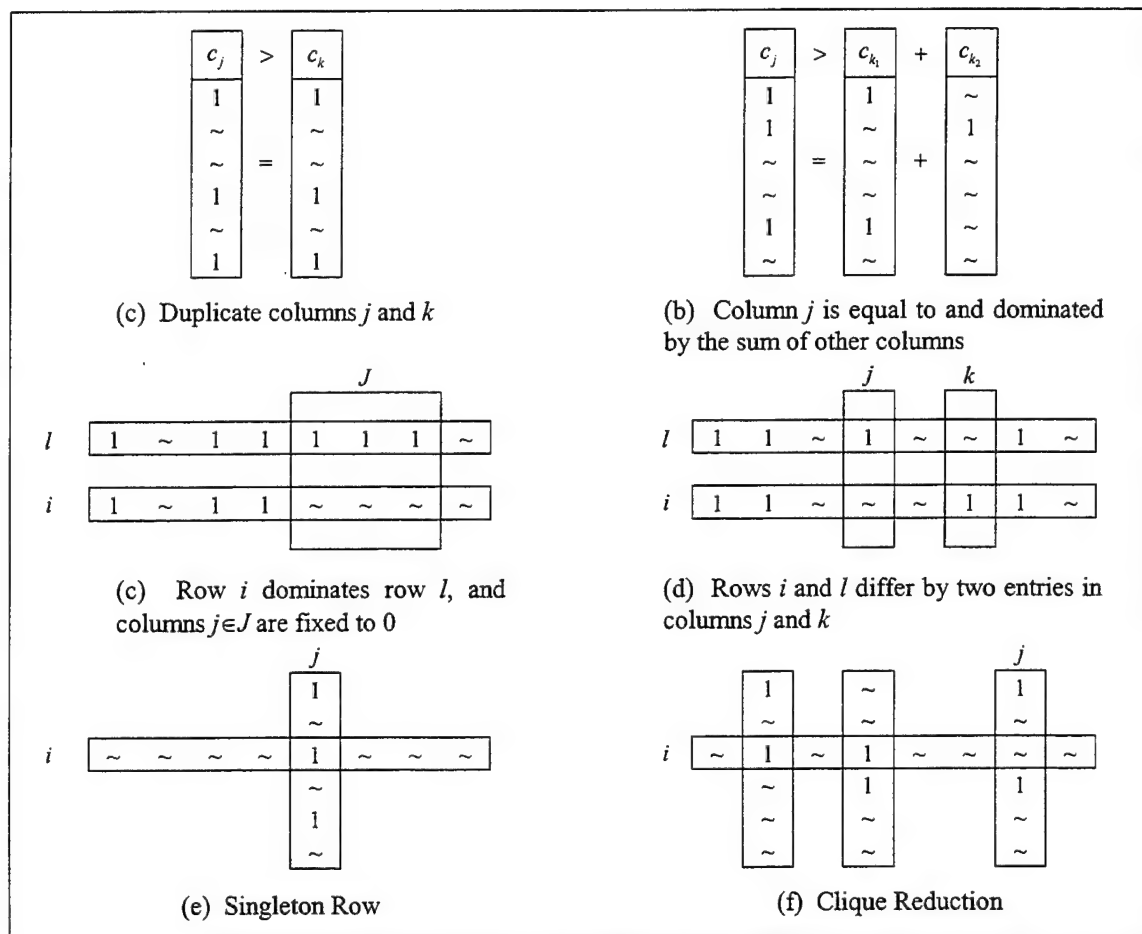
Figure V.2. SPP Reduction Rules. In Figure (a), columns j and k are the same. Column j has a higher cost. Thus, x_j is fixed to zero. In Figure (b), column j is equal to the sum of columns k_1 and k_2. The cost coefficient of column j is greater than the sum of the cost coefficients of k_1 and k_2. Hence, x_k is fixed to zero. In Figure (c), row i is a proper subset of row l. The variables that have non-zero coefficients in row l, but not in row i, are fixed to zero. In Figure (d), rows i and l share all the coefficients except the ones in columns j and k. Thus, $x_j = x_k$. In Figure (e), row i has exactly one non-zero entry at column j. Hence, x_j is fixed to one, and all incident rows are removed. In Figure (f), column j is non-orthogonal to all the columns that have non-zero entries in row i. Consequently, x_j is fixed to zero.

B.   CORRESPONDENCE OF SPP REDUCTIONS IN REFORMULATIONS

In this section, we convert SPP to its equivalent network with side constraint (*NS*)

form, and then look for reductions in the network constraints $\mathcal{N}y = b$. Any such

reduction should be equivalent to a known reduction for SPP, *or suggest a new one.*

Consider the reformulation of SPP given by (*NS*),

minimize $d'y$

(*NS*)   s.t. $\mathcal{N}y = b$

$Sy = 0$

$y \ \ binary$.

$\mathcal{N}$ is the node-arc incidence matrix of a network with one more row than SPP. The

right-hand side vector b has +1 as its first entry, is followed by zeros, and −1 as the last

entry. An arc in the network whose incident nodes are given by (f_j^k, t_j^k) represents each

variable y_j^k. That is, arc y_j^k is incident *from* node f_j^k *to* node t_j^k. Notice that, $a_{f_j^k j}^k = 1$

and $a_{t_j^k j}^k = -1$ where $a_{f_j^k j}^k$ and $a_{t_j^k j}^k$ are the non-zero coefficients of the variable y_j^k. (For

ease of exposition, each element of the set of arcs is denoted by the corresponding

variable.) The set of arcs incident to node i is denoted by

$$\mathcal{T}_i = \left\{ y_j^k \mid t_j^k = i, j \in N, k \in \left\{1, ..., |K_j|\right\} \right\}$$

and the set of arcs incident from node i is denoted by

$$\mathcal{F}_i = \left\{ y_j^k \mid f_j^k = i, j \in N, k \in \left\{1, ..., |K_j|\right\} \right\}.$$

There is no arc from the last node ($i = m+1$) to the first node ($i=1$). Thus, all entries at the first row of matrix $\mathcal{N}$ are non-negative, and all entries at the last row are non-positive. That is, $\mathcal{T}_1 = \{\}$, and $\mathcal{F}_{m+1} = \{\}$.

Observe that the variable y_j^k is obtained from the variable x_j in the original SPP. Variable y_j^k has a $+1$ coefficient at row f_j^k, and a -1 coefficient at row t_j^k. This implies that the variable x_j has consecutive ones coefficients between the rows f_j^k and $(t_j^k - 1)$ in the original SPP (i.e., $a_{f_j^k} = a_{f_j^k+1}, \ldots, = a_{t_j^k-2} = a_{t_j^k-1} = 1$).

Using the *node-arc set* notation of matrix $\mathcal{N}$, the constraints of (*SPP*) can be written as follows.

Let $\Upsilon_i = \left\{ y_j^k \mid f_j^k < i < t_j^k, j \in N, k \in \left\{1, \ldots, |K_j|\right\}\right\}$ for some row i in $\mathcal{N}$. Let

$$\mathcal{J}_i' = \left\{ j \mid y_j^k \in \Upsilon_i \text{ for some } k \in \left\{1, \ldots, |K_j|\right\}, \text{ and } j \in N \right\}$$

$$\mathcal{J}_i'' = \left\{ j \mid y_j^k \in \mathcal{F}_i \text{ for some } k \in \left\{1, \ldots, |K_j|\right\}, \text{ and } j \in N \right\}$$

$$\mathcal{J}_i''' = \left\{ j \mid y_j^k \in \mathcal{T}_i \text{ for some } k \in \left\{1, \ldots, |K_j|\right\}, \text{ and } j \in N \right\}.$$

Rows $(i-1)$ and i of the original SPP associated with some row i of $\mathcal{N}$ can be expressed, respectively, as follows:

$$\sum_{j \in \mathcal{J}_i'} x_j + \sum_{j \in \mathcal{J}_i''} x_j = 1 \qquad (i-1) \tag{V.1.a}$$

$$\sum_{j \in \mathcal{J}_i'} x_j + \sum_{j \in \mathcal{J}_i'''} x_j = 1 \qquad (i) \tag{V.1.b}$$

79

Set $\mathcal{J}_i'$ indicates the columns of (SPP) which have non-zero entries both at row i and $i-1$. Set $\mathcal{J}_i''$ indicates the columns that have non-zero entries at row i, but not at row $i-1$. Set $\mathcal{J}_i'''$ indicates the columns that have non-zero entries at row $i-1$, but not at row i.

Example V.2 illustrates the sets defined in this section, and shows the relationships between the matrix A of (SPP) and matrix $\mathcal{N}$ of (NS).

Example V.2: Consider the following SPP.

minimize $c'x$

s.t.

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

x *binary*

First, add a redundant last row $0 \cdot x = 0$ to the set of constraints. Then, perform the following elementary row operation for each $i = 5, 4, 3, 2, 1$, in the stated order: Subtract the ith constraint from the $(i+1)^{\text{th}}$ constraint. These operations create the following problem:

minimize $c'x$

s.t.

$$\begin{bmatrix}
1 & 0 & 0 & 1 & 1 & 0 & 0 & \overset{*}{1} & 1 & \overset{*}{1} & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & -1 \\
0 & 0 & -1 & -1 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & -1 & -1 & 0 \\
-1 & -1 & 0 & 0 & 0 & -1 & 0 & \underset{=}{-1} & 0 & \underset{=}{-1} & -1 & 0 & 0 & 0 & 0
\end{bmatrix} x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

x *binary*

The constraint coefficient matrix of the above formulation has exactly one $+1$ and one -1 in each column, except the distinguished columns eight and ten. When we split these columns to obtain the reformulation (*NS*), variable x_8 will be replaced with the variables y_8^1 and y_8^2, and variable x_{10} will be replaced with the variables y_{10}^1, y_{10}^2, and y_{10}^3. For all other columns, we will simply substitute x_j with y_j^1.

For the third row of matrix (*NS*), we have the following node-arc sets:

$$\Upsilon_3 = \left\{ y_1^1, y_2^1, y_3^1, y_4^1, y_9^1, y_{12}^1, y_{13}^1, y_{14}^1, y_{15}^1 \right\}, \ \mathcal{F}_3 = \left\{ y_6^1, y_7^1, y_{10}^2 \right\}, \ \mathcal{I}_3 = \left\{ y_8^1 \right\},$$

$$J_3' = \left\{ 1,2,3,4,9,12,13,14,15 \right\}, \ J_3'' = \left\{ 6,7,10 \right\}, \ \text{and } J_3''' = \left\{ 8 \right\}.$$

Observe that substituting these sets in equations (V.1.a) and (V.1.b), we obtain the second and third constraints of the given SPP.

The reduction procedures discussed in this section are based on network feasibility and variable conflicts that become apparent with flow conditions in the network with side constraints model, (*NS*).

If the variable y_j^l is fixed to either zero or one using a network reduction rule, then each variable y_j^k can also be fixed to the same value for all $k \in \{1,...,|K_j|\} \setminus l$. This is true because y_j^k is required to satisfy

$$y_j^k = y_j^{k+1}, \; k = 1,...,|K_j|-1, \text{ for each } j \in \Gamma,$$

by the side constraints $Sy = 0$. Furthermore, the variable x_j in the original SPP can also be fixed to the same value by Proposition II.1.

Next, we present the reduction rules observed in the reformulated problem, (NS). We also explore the relationships between these reductions and those that are presented in the previous section. That is, we want to know whether the reformulation model yields any different reductions in the original problem that are not detected by the known reduction rules.

1. Singleton Transshipment Row

If a *transshipment row* i (i.e., $b_i = 0$) of $\mathcal{N}$ has exactly one non-zero entry, then the variable corresponding to this entry can be fixed to the absolute value of the right-hand side coefficient.

Reduction: For a node i with $|\mathcal{F}_i| = 1$ and $|\mathcal{T}_i| = 0$, or $|\mathcal{F}_i| = 0$ and $|\mathcal{T}_i| = 1$, y_j^l is fixed to $|b_i|$ where y_j^l is the unique arc in $\mathcal{F}_i$ or $\mathcal{T}_i$.

Relationship with known reductions: Let i be a singleton transshipment row such that $|\mathcal{F}_i| = 1$, $|\mathcal{T}_i| = 0$, and y_j^l is the unique arc in $\mathcal{F}_i$. Using the above reduction rule, y_j^l can be fixed to b_i, and accordingly, x_j can also be fixed to b_i.

82

The network row i corresponds to the rows $(i-1)$ and i in SPP. As described earlier, the rows $(i-1)$ and i of SPP can be written as follows:

$$\sum_{j \in \mathcal{I}_i'} x_{\bar{j}} + \sum_{j \in \mathcal{I}_i^*} x_{\bar{j}} = 1 \qquad (i-1)$$

$$\sum_{j \in \mathcal{I}_i'} x_{\bar{j}} + \sum_{j \in \mathcal{I}_i^*} x_{\bar{j}} = 1 \qquad (i)$$

For our case (where $|\mathcal{F}_i| = 1$, $|\mathcal{T}_i| = 0$, and y_j' is the unique arc in $\mathcal{F}_i$), $\mathcal{I}_i'' = \{j\}$ and $\mathcal{I}_i''' = \{\}$. Consequently, the rows $(i-1)$ and i appear in SPP formulation as follows:

$$\sum_{j \in \mathcal{I}_i'} x_{\bar{j}} \qquad = 1 \qquad (i-1)$$

$$\sum_{j \in \mathcal{I}_i'} x_{\bar{j}} + x_j = 1 \qquad (i)$$

Now, we investigate whether or not we can still obtain the same reduction (i.e., x_j can still be fixed to b_i) after implementing the known reduction rules on these two rows.

For $i = 1$, both the first row in (NS) and SPP are singletons, and they yield the same reduction. For $2 \le i \le m$, constraints $(i-1)$ and i share all variables but x_j. Thus, using the row dominance reduction rule, x_j can be fixed to zero $(=b_i)$.

Similar arguments hold for the case where $|\mathcal{F}_i| = 0$ and $|\mathcal{T}_i| = 1$.

We therefore conclude that the outcome of this network reduction corresponds to a known SPP reduction.

83

2. A Transshipment Row Has All +1 Or All −1 Entries

If a transshipment row i has all +1 or all −1 entries, then the variables corresponding to these entries can be fixed to zero.

Reduction: For a transshipment node i, if $|\mathfrak{F}_i| \geq 1$ and $|\mathfrak{T}_i| = 0$, or $|\mathfrak{T}_i| \geq 1$ and $|\mathfrak{F}_i| = 0$, then variables y_j^k can be fixed to zero for all j, k such that $y_j^k \in \mathfrak{F}_i$ or $y_j^k \in \mathfrak{T}_i$.

Relationship with known reductions: Let $|\mathfrak{F}_i| \geq 1$ and $|\mathfrak{T}_i| = 0$. The rows $(i-1)$ and i of (*SPP*) associated with row i of (*NS*) can be written as follows:

$$\sum_{j \in \mathcal{J}_i'} x_{\bar{j}} = 1 \qquad (i-1)$$

$$\sum_{j \in \mathcal{J}_i'} x_{\bar{j}} + \sum_{j \in \mathcal{J}_i''} x_j = 1 \qquad (i)$$

Row $(i-1)$ dominates row i. Using the "dominated rows" reduction rule, variables x_j can be fixed to zero for $j \in \mathcal{J}_i''$. Observe that the network reduction rule presented in this section also yields the same reduction operations in (*SPP*).

Similar arguments hold for the case where $|\mathfrak{F}_i| = 0$ and $|\mathfrak{T}_i| \geq 1$.

3. A Transshipment Row Has Exactly One +1 And One −1 Entry

If a *transshipment row* i has exactly one +1 and one −1 entry, then the two columns associated with these entries can either be removed from the formulation (if the corresponding variables in SPP are non-orthogonal) or merged into one column (if they are orthogonal).

84

Observe that this reduction rule and the "two rows differ by two entries" reduction rule (presented in Section V.A.4) are identical in the sense that they yield the same reductions in the original (SPP) formulation.

Reduction: For a transshipment node i, if $|\mathcal{F}_i| = 1$ and $|\mathcal{T}_i| = 1$, then $y_j^{\bar{k}} = y_{\underline{j}}^k$ where $y_j^{\bar{k}} \in \mathcal{F}_i$, and $y_{\underline{j}}^k \in \mathcal{T}_i$. If $A_j' A_{\underline{j}} \geq 1$ for $\bar{j}, \underline{j} \in N$, then $y_j^{\bar{k}}, y_{\underline{j}}^k$ are both fixed to zero, else $y_j^{\bar{k}}$ and $y_{\underline{j}}^k$ are merged. In either case, the transshipment row i can be removed from the formulation (NS).

4. A Transshipment Row In (NS) Is Dominated By A Row In (SPP)

For a transshipment node i in $\mathcal{N}$ and some row $\tilde{i}$ in (SPP), if $(\mathcal{I}_i'' \cup \mathcal{I}_i''') \subseteq N_{\tilde{i}}$, then y_j^k can be fixed to zero for all j, k such that $y_j^k \in (\mathcal{F}_i \cup \mathcal{T}_i)$. (Remember that $N_{\tilde{i}}$ is the set of column indices that have non-zero entries in row $\tilde{i}$).

Justification: Constraints $(i-1)$ and i of (SPP) associated with the transshipment row i can be written as follows:

$$\sum_{j \in \mathcal{I}_i'} x_j + \sum_{j \in \mathcal{I}_i''} x_j = 1 \qquad (i-1)$$

$$\sum_{j \in \mathcal{I}_i'} x_j + \sum_{j \in \mathcal{I}_i'''} x_j = 1 \qquad (i).$$

Let $\tilde{i}$ be some row in (SPP) such that $(\mathcal{I}_i'' \cup \mathcal{I}_i''') \subseteq N_{\tilde{i}}$. Row $\tilde{i}$ is given by:

$$\sum_{j \in \mathcal{I}_i''} x_j + \sum_{j \in \mathcal{I}_i'''} x_j + \sum_{j \in N_{\tilde{i}} \setminus (\mathcal{I}_i'' \cup \mathcal{I}_i''')} x_j = 1 \qquad (\tilde{i}).$$

Columns $j \in (\mathcal{J}' \cup \mathcal{J}'' \cup \mathcal{J}''')$ are non-orthogonal and their intersection graph forms a clique. Thus, the following constraint is a valid inequality for (SPP).

$$\sum_{j \in \mathcal{J}'} x_j + \sum_{j \in \mathcal{J}''} x_j + \sum_{j \in \mathcal{J}'''} x_j \leq 1.$$

Both constraints $(i-1)$ and i dominate this valid inequality. Using the "dominated rows" reduction rule, variables x_j can be fixed to zero for $j \in (\mathcal{J}'' \cup \mathcal{J}''')$. Consequently, by Proposition II.1, the associated variables y_j^k of (NS) can also be fixed to zero for all $k = 1, ..., |K_j|$.

Example V.3:

Consider the following three constraints taken from an SPP.

$$
\begin{array}{llllllllllllll}
x_1 & +x_2 & & & & +x_6 & +x_7 & +x_8 & +x_9 & & & = & 1 & \quad (i)\\
x_1 & +x_2 & +x_3 & +x_4 & +x_5 & +x_6 & & & & & & = & 1 & \quad (ii)\\
& & +x_3 & +x_4 & +x_5 & & +x_7 & +x_8 & +x_9 & +x_{10} & x_{11} & = & 1 & \quad (iii)
\end{array}
$$

Subtracting (i) from (ii), we obtain the following transshipment row:

$$
\begin{array}{llllllll}
+x_3 & +x_4 & +x_5 & & -x_7 & -x_8 & -x_9 & = \quad 0 \qquad (ii')
\end{array}
$$

Now, consider constraints (ii') and (iii) together.

$$
\begin{array}{lllllllll}
+x_3 & +x_4 & +x_5 & & -x_7 & -x_8 & -x_9 & & = \quad 0 \qquad (ii')\\
+x_3 & +x_4 & +x_5 & & +x_7 & +x_8 & +x_9 & +x_{10} & +x_{11} = \quad 1 \qquad (iii)
\end{array}
$$

Constraint (iii) states that among the variables $x_3, x_4, x_5, x_7, x_8,$ and x_9, at most one variable is equal to one. If one of $x_3, x_4, x_5, x_7, x_8,$ or x_9 is equal to one, then another of these variables with an opposite sign must also be equal to one to satisfy constraint (ii')

(e.g., if $x_3 = 1$, then one of x_7, x_8, or x_9 must also be equal to one). This is, however, not feasible for constraint (iii). Thus, x_3, x_4, x_5, x_7, x_8, and x_9 can be fixed to zero.

The same reduction can also be obtained by applying a combination of clique and row dominance reduction rules on the three SPP constraints.

The columns 1,...,9 are non-orthogonal and their intersection graph forms a clique. Thus, the following equation is a valid inequality for this SPP.

$$x_1 \quad +x_2 \quad +x_3 \quad +x_4 \quad +x_5 \quad +x_6 \quad +x_7 \quad +x_8 \quad +x_9 \quad \leq \quad 1 \qquad \text{(iv)}$$

Constraint (i) dominates constraint (iv), and they share all variables except x_3, x_4, x_5. Therefore, these variables can be fixed to zero. Constraint (ii) also dominates constraint (iv) and they share all variables except x_7, x_8, and x_9. Hence, x_7, x_8, and x_9 can also be fixed to zero.

5. Clique Dominance

For a transshipment node i of matrix $\mathcal{N}$ and for some row $\tilde{i}$ of (SPP), if $N_{\tilde{i}} \subseteq (\mathcal{I}_i'' \cup \mathcal{I}_i''')$ and $(\mathcal{I}_i'' \cup \mathcal{I}_i''') \setminus N_{\tilde{i}} \subseteq \mathcal{I}_i''$, then variables x_j can be fixed to zero for all $j \in (\mathcal{I}_i'' \cap N_{\tilde{i}})$. Or, if $N_{\tilde{i}} \subseteq (\mathcal{I}_i'' \cup \mathcal{I}_i''')$ and $(\mathcal{I}_i'' \cup \mathcal{I}_i''') \setminus N_{\tilde{i}} \subseteq \mathcal{I}_i'''$, then variables x_j can be fixed to zero for all $j \in (\mathcal{I}_i''' \cap N_{\tilde{i}})$. Consequently, the variables y_j^k in (NS), associated with x_j, can also be fixed to zero for all $k = 1,...,|K_j|$.

Justification: Constraints $(i-1)$ and i of (SPP) associated with the transshipment row i can be written as follows:

87

$$\sum_{j \in \mathcal{I}_i'} x_j + \sum_{j \in \mathcal{I}_i''} x_j = 1 \qquad (i-1)$$

$$\sum_{j \in \mathcal{I}_i'} x_j + \sum_{j \in \mathcal{I}_i''} x_j = 1 \qquad (i).$$

Let $\tilde{i}$ be some row in (SPP) such that $N_{\tilde{i}} \subseteq (\mathcal{I}_i'' \cup \mathcal{I}_i''')$ and $(\mathcal{I}_i'' \cup \mathcal{I}_i''') \setminus N_{\tilde{i}} \subseteq \mathcal{I}_i''$. Row $\tilde{i}$ is given by:

$$\sum_{j \in \mathcal{I}_i'''} x_j + \sum_{j \in (\mathcal{I}_i'' \cap N_{\tilde{i}})} x_j = 1 \qquad (\tilde{i}).$$

Columns $j \in (\mathcal{I}_i' \cup (\mathcal{I}_i'' \cap N_{\tilde{i}}) \cup \mathcal{I}_i''')$ are non-orthogonal and their intersection graph forms a clique. Thus, the following constraint is a valid inequality for (SPP).

$$\sum_{j \in \mathcal{I}_i'} x_j + \sum_{j \in \mathcal{I}_i''} x_j + \sum_{j \in (\mathcal{I}_i'' \cap N_{\tilde{i}})} x_j \leq 1.$$

The valid inequality is dominated by the constraint $(i-1)$. Using the "dominated rows" reduction rule, variables x_j can be fixed to zero for $j \in (\mathcal{I}_i'' \cap N_{\tilde{i}})$. Consequently, by Proposition II.1, the associated variables y_j^k of (NS) can also be fixed to zero for all

$k = 1, ..., |K_j|$.

After this reduction, the constraints $(i-1)$ and $\tilde{i}$ become identical, and one of them can be removed from the formulation (SPP).

Example V.4:

Consider the following three constraints taken from an SPP.

$$
\begin{array}{llllllllllll}
x_1 & +x_2 & & & & +x_6 & +x_7 & +x_8 & +x_9 & = & 1 & \qquad \text{(i)} \\
x_1 & +x_2 & +x_3 & +x_4 & +x_5 & +x_6 & & & & = & 1 & \qquad \text{(ii)} \\
& & & & +x_5 & & +x_7 & +x_8 & +x_9 & = & 1 & \qquad \text{(iii)}
\end{array}
$$

88

Subtracting (i) from (ii), we obtain the following transshipment row.

$$+x_3 \quad +x_4 \quad +x_5 \qquad -x_7 \quad -x_8 \quad -x_9 \quad = \quad 0 \qquad \text{(ii}')$$

Now, consider constraints (ii′) and (iii) together.

$$+x_3 \quad +x_4 \quad +x_5 \qquad -x_7 \quad -x_8 \quad -x_9 \quad = \quad 0 \qquad \text{(ii}')$$
$$+x_5 \qquad +x_7 \quad +x_8 \quad +x_9 \quad = \quad 1 \qquad \text{(iii)}$$

If x_5 is equal to one, then one of the variables x_7, x_8, and x_9 must also be equal to one to satisfy constraint (ii′). This is, however, not feasible for constraint (iii). Thus, x_5 is fixed to zero.

The same reduction can also be obtained by applying a combination of clique and row dominance reduction rules to the three SPP constraints.

The columns 1,2,5,6,7,8, and 9 are non-orthogonal and their intersection graph forms a clique. Thus, the following equation is a valid inequality for this SPP.

$$x_1 \quad +x_2 \qquad +x_5 \quad +x_6 \quad +x_7 \quad +x_8 \quad +x_9 \quad \leq \quad 1 \qquad \text{(iv)}$$

Constraint (i) dominates constraint (iv) and they share all variables except x_5. Therefore, x_5 can be fixed to zero.

Figure V.3 illustrates the reductions described in this section.

(a) Singleton transshipment row i

(b) Transshipment row i has all +1 entries in columns j, k, and l

(c) Transshipment row i has exactly one +1 in column j and one −1 in column k

(d) Transshipment row i in (NS) is dominated by row $\tilde{\imath}$ in (SPP)

(e) Clique dominance reduction resulting from row i in (NS) and row $\tilde{\imath}$ in (SPP)
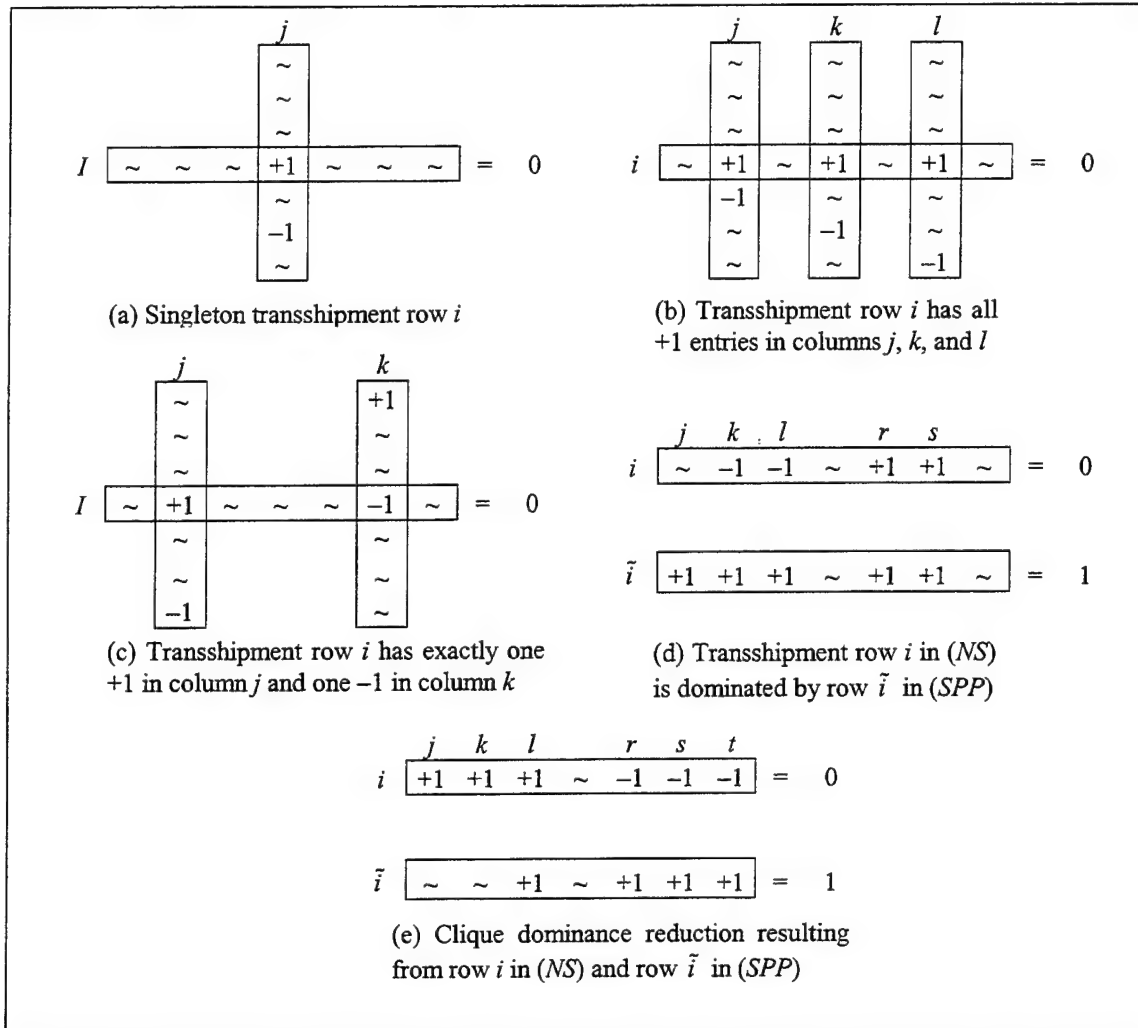
Figure V.3. Network Reduction Rules. In Figure (a), row i has exactly one non-zero entry at column j. Hence, x_j is fixed to zero. In Figure (b), row i has all +1 entries at columns j, k, and l. Thus, x_j, x_k, and x_l are fixed to zero. In Figure (c), row i has exactly one +1 entry at column j, and one −1 entry at column k. Hence, $x_j = x_k$. If columns j and k are non-orthogonal, then x_j and x_k are fixed to zero. Otherwise, columns j and k are merged. In Figure (d), for every non-zero entry in transshipment row i, there corresponds a non-zero entry in (SPP) row $\tilde{\imath}$. Variables x_k, x_l, x_r, and x_s are fixed to zero. Consequently, x_j is fixed to one. In Figure (e), x_l is fixed to zero. Because, if x_l is equal to one, then one of the variables x_r, x_s, or x_t must also be equal to one to satisfy constraint i. However, this solution does not satisfy constraint $\tilde{\imath}$.

90

C. USE OF HIDDEN NETWORK STRUCTURE

By finding a hidden network row submatrix of matrix A, (SPP) can be transformed to a network flow problem with side constraints.

Let E_1 be an $(m_1 \times n)$ row submatrix of A that is transformable to a network, and let E_2 denote the submatrix involving the rest of the rows of A. Thus, (SPP) can be represented as:

minimize $c'x$

    s.t. $E_1 x = e_1$

            $E_2 x = e_2$

             x binary

where e_1 and e_2 are conformable vectors consisting of all ones. Let T be a transformation matrix of size $(m_1 + 1) \times m_1$. That is, pre-multiplying both sides of $E_1 x = e_1$ by T, we obtain pure network constraints. The resulting reformulation of (SPP) is given by:

        minimize $c'x$

(HN) s.t. $\mathcal{N}x = r$

             $E_2 x = e_2$

             x binary

where $\mathcal{N}$ is an $(m_1 + 1) \times n$ node-arc incidence matrix corresponding to a network, r is the integral vector of requirements at $(m_1 + 1)$ nodes in the network.

Ali et al. [1995] suggest heuristic procedures that may (with some mathematical repairs to their presentation) reveal a row submatrix of SPP that is transformable to a network. Then, they formulate SPP as a pure network flow problem with side constraints, and present some reduction rules for the original SPP based on the flow conditions in the network.

The reduction rules suggested by Ali et al. are similar to those presented in the previous section: (1) singleton network row, (2) a transshipment row has exactly one +1 and one −1 entry, and (3) a transshipment row in (*NS*) is dominated by a transshipment row in (*SPP*). The difference between the reductions suggested by Ali et al. and these three reductions is that the former are implemented on the reformulation (*HN*), and the latter are implemented on (*NS*). The other two reduction rules presented in the previous section (i.e., (1) a transshipment row has all +1 or all −1 entries, and (2) clique dominance) can also be implemented on (*HN*) in addition to those suggested by Ali et al..

In the previous section, we have shown that the outcome of each network reduction can also be detected by a combination of known SPP reduction rules. Using similar arguments, we see that the reductions that Ali et al. present as new can actually be detected by a combination of known SPP reduction rules presented in Section A.

Next, we introduce a new method to obtain additional reductions in SPP. This method originated while we were experimenting with the relationship between the reductions observed in the reformulations and the known SPP reductions.

D.   A NEW SPP REDUCTION METHOD: COLUMN SPLIT REDUCTION

1.   Generating Valid Equalities That Yield More New Clique Reductions for SPP

Let i and k be two constraints in *(SPP)*. Let $\mathcal{J}_i$ denote the set of column indices that have non-zero entries in row i, but not in row k. Let $\mathcal{J}_k$ denote the set of column indices that have non-zero entries in row k, but not in row i. And, let $\mathcal{J}_{ik}$ denote the set of column indices that have non-zero entries in both rows. Thus, constraints i and k can be written as:

$$\sum_{j \in \mathcal{J}_i} x_j + \sum_{j \in \mathcal{J}_{ik}} x_j = 1 \qquad\qquad (i)$$

$$\sum_{j \in \mathcal{J}_k} x_j + \sum_{j \in \mathcal{J}_{ik}} x_j = 1 \qquad\qquad (k)$$

Proposition V.1: For some constraint l of *(SPP)*, if $N_l \subset (\mathcal{J}_i \cup \mathcal{J}_k \cup \mathcal{J}_{ik})$, $(N_l \cap \mathcal{J}_i) \subset \mathcal{J}_i$, and $(N_l \cap \mathcal{J}_k) \subset \mathcal{J}_k$, then the following constraint is a *valid equality* (i.e., this equality constraint is satisfied by all integer solutions to *(SPP)*):

$$\sum_{j \in (\mathcal{J}_i \backslash N_l)} x_j + \sum_{j \in (\mathcal{J}_{ik} \cap N_l)} x_j + \sum_{j \in (\mathcal{J}_k \backslash N_l)} x_j = 1 \qquad\qquad (V.2)$$

Proof: Columns $j \in (\mathcal{J}_{i,k} \backslash N_l) \cup N_l$ are non-orthogonal, and their intersection graph forms a clique. Thus, the following constraint is a valid inequality for *(SPP)*.

$$\sum_{j \in (\mathcal{J}_{ik} \backslash N_l)} x_j + \sum_{j \in N_l} x_j \leq 1.$$

Evidently, this valid inequality is dominated by row l. Using the "dominated rows" reduction rule, the variables x_j can be fixed to zero for $j \in (\mathcal{J}_{ik} \backslash N_l)$.

93

Sets $\mathcal{J}_i$ and $\mathcal{J}_k$ can be expressed as $(\mathcal{J}_i \setminus N_l) \cup (\mathcal{J}_i \cap N_l)$ and $(\mathcal{J}_k \setminus N_l) \cup (\mathcal{J}_k \cap N_l)$, respectively. By implementing the variable reductions above, and replacing $\mathcal{J}_i$ and $\mathcal{J}_k$ with the new expressions, constraints i and k can be written as follows:

$$\sum_{j \in (\mathcal{J}_i \setminus N_l)} x_j + \sum_{j \in (\mathcal{J}_i \cap N_l)} x_j + \sum_{j \in (\mathcal{J}_{ik} \cap N_l)} x_j = 1 \qquad (i)$$

$$\sum_{j \in (\mathcal{J}_k \setminus N_l)} x_j + \sum_{j \in (\mathcal{J}_k \cap N_l)} x_j + \sum_{j \in (\mathcal{J}_{ik} \cap N_l)} x_j = 1 \qquad (k).$$

Set N_l can be expressed as the union of three disjoint subsets: (1) $(\mathcal{J}_i \cap N_l)$, (2) $(\mathcal{J}_k \cap N_l)$, and (3) $(\mathcal{J}_{ik} \cap N_l)$. Thus, constraint l can be written as follows:

$$\sum_{j \in (\mathcal{J}_i \cap N_l)} x_j + \sum_{j \in (\mathcal{J}_k \cap N_l)} x_j + \sum_{j \in (\mathcal{J}_{ik} \cap N_l)} x_j = 1 \qquad (l).$$

To satisfy constraint (l), exactly one of the variables x_j must be equal to one for $j \in N_l$. For the three disjoint subsets of N_l, we have the following three cases to consider:

Case (1): If $x_j = 1$ for some $j \in (\mathcal{J}_i \cap N_l)$, then constraint i is satisfied. To satisfy constraint k, exactly one of the variables x_j must be equal to one for $j \in (\mathcal{J}_k \setminus N_l)$.

Case (2): If $x_j = 1$ for some $j \in (\mathcal{J}_k \cap N_l)$, then constraint k is satisfied. To satisfy constraint i, exactly one of the variables x_j must be equal to one for $j \in (\mathcal{J}_i \setminus N_l)$.

Case (3): If $x_j = 1$ for some $j \in (\mathcal{J}_{ik} \cap N_l)$, then all three constraints i, k, and l are satisfied.

94

Exactly one of the three disjoint cases must occur. This implies that exactly one of the variables x_j must be equal to one for $j \in (\mathcal{J}_i \setminus N_l)$, $j \in (\mathcal{J}_k \setminus N_l)$, or $j \in (\mathcal{J}_{ik} \cap N_l)$ (i.e., $j \in (\mathcal{J}_i \setminus N_l) \cup (\mathcal{J}_k \setminus N_l) \cup (\mathcal{J}_{ik} \cap N_l)$). Thus, Equation V.2 is a valid equality for (*SPP*). Q.E.D.

Equation V.2 states that each pair of columns $A_{j'}$ and $A_{j''}$ are non-orthogonal for $j' \in (\mathcal{J}_i \setminus N_l)$, and $j'' \in (\mathcal{J}_k \setminus N_l)$. This relationship is not obvious if we do not generate the valid equality given by Equation V.2.

Recall that when we construct the intersection graph $G_A = (N, \mathbb{E})$ of (*SPP*), we define a node $j \in N = \{1, 2, ..., n\}$ for each column A_j of A, and join two nodes $i \ne j \in N$ by an edge $(i, j) \in \mathbb{E}$ if the columns A_i and A_j of A are non-orthogonal. Given the subgraph of G_A constructed by the constraints i, k, and l only, there does not exist an arc between a node $j' \in (\mathcal{J}_i \setminus N_l)$ and a node $j'' \in (\mathcal{J}_k \setminus N_l)$. This is because columns $A_{j'}$ and $A_{j''}$ have no common entry equal to +1 in rows i, k, and l. However, when we add the valid equality obtained from constraints i, k, and l to the formulation (*SPP*), both columns $A_{j'}$ and $A_{j''}$ will have +1 entries in common in this valid equality. Consequently, we can add a new arc to the intersection graph G_A for each pair of nodes j' and j'', such that $j' \in (\mathcal{J}_i \setminus N_l)$ and $j'' \in (\mathcal{J}_k \setminus N_l)$.

By extending the clique associated with the variables that have non-zero entries in a valid equality, we may find new "clique reductions" for SPP. That is, if a column is non-orthogonal to all columns with non-zero entries in a valid equality, then this column

95

is fixed to zero. Note that after we fix some variables as a result of these new clique reductions, some other variable and/or row reductions may also become apparent in (SPP).

The worst-case complexity of finding valid equalities using this new method is $O(m^2 \times l)$, where m is the number of constraints and l is the number of non-zero entries in matrix A. We choose a pair of rows from matrix A, and compare the non-zero entries in these rows with the non-zero entries in the other rows. For each pair of rows, we make at most l comparisons. The number of possible row pair combinations is:

$$\binom{m}{2} = \frac{m^2 - m}{2}.$$

Thus, the number of comparisons is at most $\dfrac{m^2 l - ml}{2}$, and the worst-case complexity is $O(m^2 \times l)$.

As stated in Section A.6, the worst-case complexity of finding clique reductions in an SPP is $O(m \times n \times l)$. Assuming that the number of columns is greater than the number of rows, the worst-case complexity of extending the intersection graph using the new method and finding clique reductions in the extended graph is still $O(m \times n \times l)$.

When we implement the comparisons to detect the valid equalities, we can also detect the "dominated rows" and "two rows differ by two entries" reductions. The network reductions described in Sections V.B.4 and V.B.5 can also be detected during these comparisons. Recall that the outcomes of these network reductions correspond to combinations of "dominated rows" and "clique" reductions.

Example V.5: Consider the following three constraints taken from an SPP.

$$x_1 \qquad\qquad +x_6 \;\; +x_7 \;\; +x_8 \;\; +x_9 \;\; +x_{10} \;\; +x_{11} \;\; = \;\; 1 \qquad\qquad (i)$$

$$x_1 \;\; +x_2 \;\; +x_3 \;\; +x_4 \;\; +x_5 \;\; +x_6 \qquad\qquad +x_9 \qquad\qquad = \;\; 1 \qquad\qquad (k)$$

$$+x_4 \;\; +x_5 \qquad\qquad +x_9 \;\; +x_{10} \;\; +x_{11} \;\; = \;\; 1 \qquad\qquad (l)$$

$$\mathcal{I}_i = \{7,8,10,11\}, \; \mathcal{I}_k = \{2,3,4,5\}, \; \mathcal{I}_{ik} = \{1,6,9\}, \text{ and } N_l = \{4,5,9,10,11\};$$

$$N_l \cap \mathcal{I}_i = \{9,10,11\}, \; N_l \cap \mathcal{I}_k = \{4,5,9\},$$

$$N_l \subset (\mathcal{I}_i \cup \mathcal{I}_k \cup \mathcal{I}_{ik}), \; (N_l \cap \mathcal{I}_i) \subset \mathcal{I}_i, \; (N_l \cap \mathcal{I}_k) \subset \mathcal{I}_k,$$

$$\mathcal{I}_i \setminus N_l = \{7,8\}, \; \mathcal{I}_k \setminus N_l = \{2,3\}, \text{ and } \mathcal{I}_{ik} \cap N_l = \{9\}.$$

Thus, by Proposition V.1 the following equation is a valid equality for the given SPP.

$$+x_2 \;\; +x_3 \qquad\qquad +x_7 \;\; +x_8 \;\; +x_9 \qquad\qquad = \;\; 1 \qquad\qquad (v)$$

If x_2 or x_3 is equal to one, then x_1, x_4, x_5, x_6, and x_9 are equal to zero. Hence, to satisfy constraint l, either x_{10} or x_{11} must be equal to one. This implies x_7 and x_8 must be equal to zero to satisfy constraint i.

If x_7 or x_8 is equal to one, then x_1, x_6, x_9, x_{10}, and x_{11} are equal to zero. Hence, to satisfy constraint l, either x_4 or x_5 must be equal to one. This implies x_2 and x_3 must be equal to zero to satisfy constraint k.

If x_9 is equal to one, then all the other variables that have a non-zero entry in constraints i, k, and l are equal to zero.

For the case where all variables x_2, x_3, x_7, x_8, and x_9 of the valid equality v are equal to zero, the given SPP is infeasible.

97

Consequently, we have justified that constraint v is satisfied for all integer solutions to the given SPP, and therefore is a valid equality.

2. Computational Results For The Column Split Reduction

We test the column split reduction on a subset of real-world airline crew scheduling problems that are also used by Hoffman and Padberg [1993]. The test data is obtained from the online OR-Library [2000] presented by J.E. Beasley.

Table V.1 shows our computational results on twelve sample problems. Unlike the algorithms presented in chapters III and IV for solving SPPs with special consecutive ones structures, the column split reduction can be applied to any SPP. Consequently, in this chapter we test the column split reduction with a wider diversity of airline crew scheduling samples.

We report the maximal number of clique reductions obtained at the root node of the B&B tree, and the maximal number of clique reductions obtained when the new method is introduced.

These reductions are implemented with Compaq Visual Fortran [1999] on a Pentium III 700Mhz personal computer with 1 Gb RAM. The largest problem SPPUS02 takes about three minutes to presolve with or without the new method. The small problems take less than a second. Here, as in any commercial solver, efficiency dictates that we carefully implement the column split reduction and take care to avoid extensive presolve times. We must balance presolve effort with the desirable reductions that accrue.

For test data SPPUS02, the column split reduction method yields a 38.4% increase in the number of variables fixed. However, for test data SPP03, the percentage increase is only 0.4%.

Problem Name	Number Of Rows	Number of Columns	Number of Non-zeros	Number of Original Clique Reductions	% of Variables Fixed Before Adding Arcs	Number of Arcs Added to the Intersection Graph	Number of Clique Reductions After Adding New Arcs	% of Variables Fixed After Adding Arcs	Number of Additional Variables Fixed	% of Additional Variables Fixed
SPPAA01	823	8,904	72,965	1,095	12.30	7,876,877	1,533	17.22	438	4.92
SPPAA02	531	5,198	36,359	1,015	19.52	50,949,975	1,328	25.55	313	6.03
SPPAA03	825	8,627	70,806	1,614	18.71	10,321,196	1,964	22.77	350	4.06
SPPAA04	426	7,195	52,121	780	10.84	4,563,495	1,075	14.95	295	4.11
SPPAA05	801	8,308	65,953	1,544	18.58	10,300,739	1,959	23.58	415	5.00
SPPAA06	646	7,292	51,728	1,074	14.73	5,955,250	1,425	19.54	351	4.81
SPPNW23	19	711	3,350	93	13.08	6,097	105	14.76	12	1.68
SPPNW26	23	771	4,215	123	15.95	2,390	130	16.86	7	0.91
SPPNW28	18	1,210	8,555	379	31.32	6,455	384	31.73	5	0.41
SPPNW35	23	1,709	10,494	256	14.97	18,810	310	18.14	54	3.17
SPPNW36	20	1,783	13,160	263	14.75	12,004	376	21.09	113	6.34
SPPUS02	100	13,635	192,716	100	0.73	34,632,043	5,339	39.16	5,239	38.43

Table V.1. Computational Results of the Column Split Reduction. The first column shows the name of each sample problem as it appears in the online reference OR-Library [2000]. The numbers of clique reductions obtained before and after implementing the new method are displayed as well as the percentage of variables fixed using the clique reduction rule. The last column displays the percentage of improvement achieved in the number of reductions using the new method.

Our results show that the column split reduction can significantly increase the number of reductions in an SPP. Moreover, we conjecture that finding such reductions will improve the solution time and/or allow some unsolvable real-world SPP applications to be solved (e.g., Hoffman and Padberg [1993, pp. 665-666]).

E. EXTRACTING HIDDEN ARCS OF THE INTERSECTION GRAPH BY PROBING

Another way of extending the intersection graph G_A with the addition of new arcs is fixing variables by probing. Probing has been used effectively in general mixed 0-1 integer programming (e.g., [Crowder et al. 1982], [Savelsbergh 1994], and [Atamturk et al. 1995]). Probing a binary variable means tentatively setting the variable to one of its bounds and observing the implications of this setting. For instance, if an infeasibility occurs when the variable is fixed to one of its bounds, then the binary variable can be fixed to the opposite bound. To the best of our knowledge, probing has not been used to extend the intersection graph G_A. In this section, we demonstrate but do not implement how we can extend the intersection graph G_A with the addition of new arcs by probing variables.

If we fix a variable x_j to one, all the incident rows are satisfied and can be removed. Moreover, all other columns incident with any of these rows can be fixed to zero. After removing all the incident rows and fixing the columns incident with these rows to zero, a second pass through the resulting smaller matrix is initiated if any additional variables are fixed using the known reduction operations. Let x_l be the variable that is fixed to zero during the second pass. Then, the following constraint is a valid inequality for (*SPP*):

$$x_j + x_l \leq 1$$

Note that columns j and l are orthogonal in the original A matrix, and there does not exist an arc between nodes j and l in the intersection graph G_A. However, the valid

100

inequality above allows us to connect nodes j and l with an arc in the intersection graph G_A. We call this a hidden arc.

In Example V.5, if we fix x_2 to one, constraint k is removed and x_1, x_3, x_4, x_5, x_6, and x_9 are fixed to zero. After these reductions, constraints i and k can be written as:

$$x_7 + x_8 + x_{10} + x_{11} = 1 \qquad (i)$$

$$x_{10} + x_{11} = 1 \qquad (l)$$

Because constraint i is dominated by constraint l, x_7 and x_8 can be fixed to zero. Therefore, if x_2 is equal to one, then x_7 and x_8 are equal to zero.

On the other hand, if x_2 is fixed to zero, then either one of x_7 or x_8 is equal to one, or they are both equal to zero. Hence, the following constraint is a valid inequality for the given SPP:

$$x_2 + \qquad\qquad + x_7 + x_8 \le 1 .$$

The node corresponding to x_2 can be connected to nodes 7, and 8. Similarly, by probing x_3, we can connect node 3 to nodes 7, and 8.

By probing each variable in an SPP as discussed above, we can extract all the hidden arcs of the intersection graph. The number of hidden arcs that can be extracted in the intersection graph by probing is at least the number of arcs that can be extracted using the row comparison technique, described in Section V.D.1.

The worst-case complexity of naively finding all the hidden arcs of the intersection graph by probing is $O(n^2 \times m \times l)$. Each variable x_j is fixed to one, all the

101

incident rows are removed, and all the other columns incident with any of these rows are fixed to zero. A second pass through the resulting smaller matrix is initiated if any additional variables are fixed using the known reduction rules. This is done in $O(n \times m \times l)$ for a single variable, because the worst-case complexity of finding clique reductions is $O(n \times m \times l)$. Hence, for all variables the worst-case complexity is $O(n^2 \times m \times l)$.

Note that extending the intersection graph G_A with the addition of new arcs generated by probing may offer new valid inequalities for SPP (e.g., *clique or odd cycle constraints* discussed in [Hoffman and Padberg 1993]). However, extracting the hidden arcs of the intersection graph by probing, and finding valid inequalities in the extended intersection graph may be computationally very expensive.

VI.   A NEW FORMULATION FOR A LONG-TERM AIRCRAFT CARRIER DEPLOYMENT SCHEDULING PROBLEM

A.   BACKGROUND

Forward deployment of Navy aircraft carrier battle groups and amphibious ready groups is a primary means for U.S. to advance overseas interests. As Department of the Navy, Naval Doctrine Publication 1 [NDP 1, 1994] states: "Overseas presence promotes national influence and access to global areas, builds regional coalitions and collective security, furthers stability, deters aggression, and provides initial crisis-response capability."

Aircraft carriers are sovereign U.S. territories that navigate anywhere in international waters (more than 70% of the earth's surface is ocean). This fact is not overlooked by those U.S. officials who make political and strategic decisions to use naval aircraft carriers as a powerful instrument of diplomacy to strengthen alliances and respond to potential and developing crises. As President Bill Clinton said during a recent visit to the aircraft carrier USS Theodore Roosevelt, "When word of crisis breaks out in Washington, it's no accident the first question that comes to everyone's lips is: where is the nearest carrier?" [U.S. Navy 1998]

At present the Navy attempts to maintain the forward presence of aircraft carriers in three Areas of Responsibility (AORs): the Mediterranean under the European Command (EUCOM), the Indian Ocean under the Central Command (CENTCOM), and in the Western Pacific under Pacific Command (PACOM). Carriers from the Atlantic Fleet (LANTFLT) fulfill forward presence requirements for the EUCOM (Mediterranean

103

Sea) AOR. Pacific Fleet (PACFLT) carriers provide coverage for the CENTCOM (Indian Ocean) and PACOM AORs. Occasionally, an Atlantic Fleet carrier will also assist in covering the Indian Ocean. Finally, a PACOM carrier operating from Yokosuka, Japan, is usually responsible for the Western Pacific.

Over the past decade or so, the Navy has tried to maintain a continuous forward carrier presence in these principal AORs. Diminishing defense budgets have limited the number of carriers available to meet this goal. Carrier availability is further constrained by scheduled maintenance, training requirements, and Chief of Naval Operation (CNO) policy on Personnel Tempo of Operations (PERSTEMPO). These restrictions, along with limited available assets, have made continuous carrier *coverage* (i.e., the percentage of time an AOR is covered by at least one carrier) essentially impossible.

Providing a sufficient amount of coverage in the AORs through forward presence helps to decrease *crisis response time*. "Crisis response, the timely dispatch of naval forces to a specific area, allows the U.S. to render assistance or exert military force." [Department of the Navy 1994, pg. 20] Herein, crisis response time is defined as the expected time for the closest carrier to arrive at a crisis location.

A new nuclear powered carrier costs about five billion dollars, its aircraft cost at least as much, and when deployed it is manned by 3,200 ship's company and 2,480 air wing personnel. The air wing consists of eight to nine squadrons (85 aircraft). A carrier operates as the centerpiece of a carrier battle group. A carrier battle group, commanded by a flag officer, normally consists of two guided missile cruisers, a guided missile destroyer, a destroyer, a frigate, two attack submarines, and a combined ammunition,

oiler, and supply ship (AOE). Figure VI.1 shows USS Nimitz (CVN 68), a nuclear powered aircraft carrier in PACFLT.
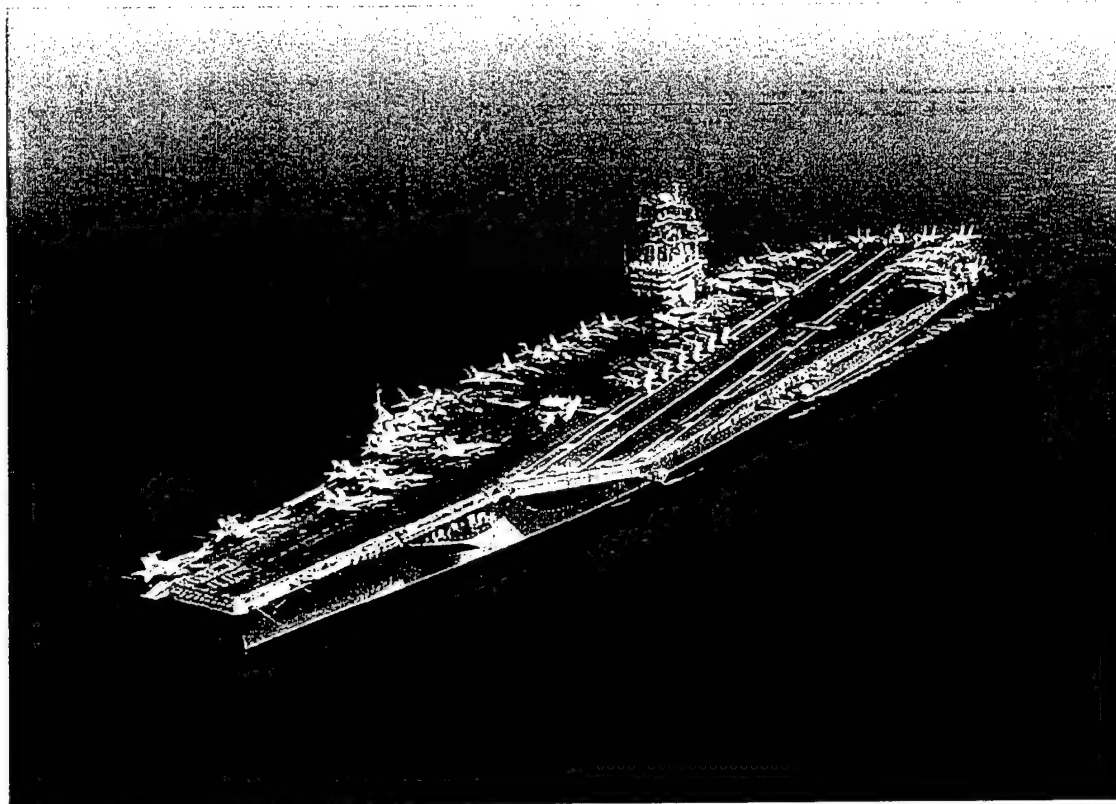
Figure VI.1. USS Nimitz (CVN 68), launched in 1975, has 3,200 crew in the ship's company and 2,480 for the air wing. The mission of Nimitz and her embarked air wing is to conduct sustained combat air operations. The air wing consists of eight to nine squadrons (85 aircraft). The ship normally operates as the centerpiece of a carrier battle group. The carrier battle group, commanded by a flag officer, consists of two guided missile cruisers, a guided missile destroyer, a destroyer, a frigate, two attack submarines, and a combined ammunition, oiler, and supply ship. [U.S. Navy 1998]

The objective of the U.S. Navy aircraft carrier scheduling problem is to minimize the number of uncovered periods across all AORs, subject to the constraints on the

number of available carriers, scheduled maintenance, training requirements, and CNO policy on PERSTEMPO, for a planning horizon of ten years.

To date, there have been two optimization-based approaches suggested for solving the U.S. Navy aircraft carrier scheduling problem. The first one is the classical set partitioning model introduced in Chapter I, Section A.3. The other one is a *two-commodity network flow model with side constraints*, which is also an integer program.

The two-commodity network flow model with side constraints is introduced by Schauppner [1996], and used by Brown et al. [1997] in an optimization-based model called the Coverage and Response Estimation (CoRE). CoRE is developed to estimate the level of forward presence sustainable by various numbers of carriers. CoRE honors pre-determined, exogenous, fixed scheduled maintenance periods for each carrier, which are stipulated by a long-range schedule published by the Planning and Engineering for Repairs and Alterations Activity for the Aircraft Carriers (PERA CV). CoRE schedules the Navy's actual carriers for deployment around these fixed periods of availability in order to maximize coverage in the AORs.

Ayik [1998] generalizes CoRE by incorporating the synchronous planning of deployments and major maintenance availabilities. Ayik's formulation is also a two-commodity network flow model with side constraints.

This chapter presents a new integer programming formulation for the long-term aircraft carrier scheduling problem. We first describe the scheduling factors and operations constraints. Next, we present the previously suggested model, a two-

106

commodity network flow problem with side constraints. Then, we introduce the new formulation and compare it with the previous models.

B. AIRCRAFT CARRIER DEPLOYMENT SCHEDULING FACTORS AND OPERATIONS CONSTRAINTS

The deployment scheduling of carriers depends on five factors:

(i)   depot level maintenance,

(ii)   work-up cycle,

(iii)   Personnel Tempo of Operations (PERSTEMPO),

(iv)   transit time, and

(v)   availability of LANTFLT carriers for CENTCOM.

Each of these factors is described below.

1.   Depot Level Maintenance

Depot level maintenance is defined as "that maintenance which requires skills or facilities beyond those of the organizational and intermediate levels and is performed by naval shipyards, naval ship facilities, or item depot activities" [OPNAV 1992]. While at depots, carriers undergo large-scale maintenance, repairs, approved alterations, and modifications to update and improve the carrier's technical and military capabilities. Each carrier periodically requires maintenance of differing durations. In general, these maintenance periods are for

(i)   incremental maintenance lasting approximately six months,

(ii)   incremental maintenance requiring dry docking, which lasts approximately twelve months, or

(iii) complex overhaul and possibly nuclear refueling, with a duration exceeding two years.

U.S. Navy ships accomplish depot maintenance at notional *intervals, durations,* and *repair man-days* set forth in OPNAVNOTE 4700 [OPNAV 1996b, pg. 3]. "Interval is defined as the period from the completion of one scheduled depot availability to the start of the next scheduled depot availability. Duration is defined as the period from the start of an availability to its completion. Repair man-days are those Type Commander maintenance man-days typically accomplished by the executing activity to satisfactorily complete the type of availability indicated."

A sample notional depot maintenance cycle for a Nimitz class aircraft carrier is provided in Figure VI.2:
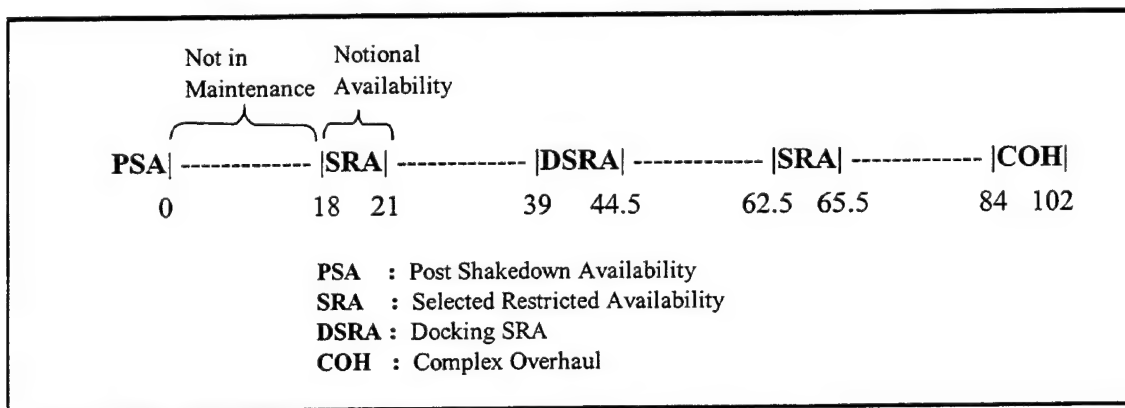
Figure VI.2. Sample Notional Depot Maintenance Availability for a Nimitz Class (CVN 68) Aircraft Carrier. The dashed time lines indicate periods not in maintenance. The time line numbers indicate months (but not to scale). A Post Shakedown Availability (PSA) may require only a few months, while a Complex Overhaul (COH) may take years.

To ensure compatibility between a ship's employment schedules and depot workloads, CNO authorizes deviation from the notional depot availability interval as shown in Table VI.1.

Months from Start of Maintenance Cycle to Start of Maintenance Period	Allowable Months Deviation of Start of Maintenance Period
0-36 mo	+/- 3 mo
37-48 mo	+/- 4 mo
49-60 mo	+/- 5 mo
61-72 mo	+/- 6 mo
73-84 mo	+/- 7 mo
>84 mo	+/- 7 mo

Table VI.1. CNO Guidelines for Altering Scheduled Maintenance Periods. During a maintenance cycle, each scheduled maintenance period may be shifted forward, or backward by a number of months increasing as we progress into the far future. A *maintenance cycle* starts after the completion of a carrier's overhaul (or docking availability, when no overhaul availabilities are included in the maintenance plan) and ends after completion of the next overhaul or docking availability. For new construction ships, the maintenance cycle starts after completion of the post shakedown availability. [OPNAV 1996b, pp. 3-4]

Figure VI.3 shows the allowable deviation durations corresponding to the notional depot maintenance availabilities, provided in Figure VI.2, for a Nimitz Class (CVN-68) aircraft carrier.

```
Start of              Start of
Maintenance           Notional
Cycle                 Availability
  │                     │
  ▼                     ▼
PSA| ------------ |SRA| ------------- |DSRA| ------------ |SRA| ------------ |COH|
  0                18   21           39    44.5        62.5   65.5        84   102
                   └──┬──┘           └───┬───┘         └───┬───┘          └──┬──┘
Allowable
Deviation:          +/- 3 mo           +/- 4 mo          +/- 6 mo          +/- 7 mo
```
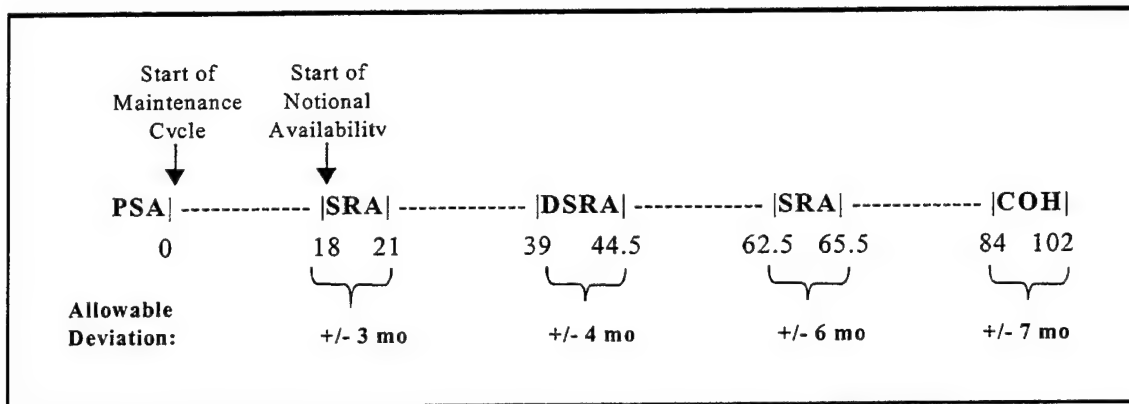
Figure VI.3. Allowable Deviations Corresponding to the Notional Depot Maintenance Availabilities for the Nimitz Class (CVN 68). This maintenance cycle begins at month 0. Notional start time for the first scheduled maintenance, a Selected Restricted Availability (SRA), is 18 months from the beginning of the maintenance cycle, and can be shifted forward or delayed by up to three months. The time line is not to scale.

Depot maintenance of aircraft carriers is conducted at four major repair facilities:

(i)   Puget Sound Naval Shipyard (PUGET),

(ii)   Norfolk Naval Shipyard (NORVA) (Figure VI.4),

(iii)   Yokosuka Ship Repair Facility (YOKO), and

(iv)   Newport News Shipbuilding Company (NEWS).
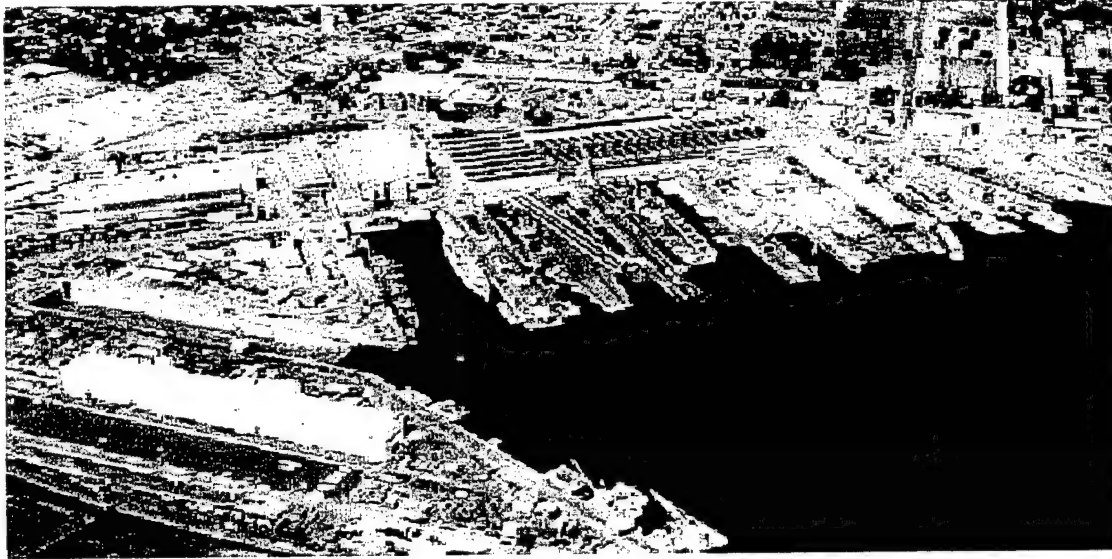
Figure VI.4.   Norfolk Naval Shipyard Aerial View. The NORFOLK NAVAL SHIPYARD in Portsmouth, Virginia, is one of the largest shipyards in the world specializing in repairing, overhauling and modernizing ships and submarines. It is the oldest and largest industrial facility belonging to the U.S. Navy, and is also the most multifaceted. At the extreme left-center is an empty dry dock that can accommodate an aircraft carrier.

Scheduling depot maintenance availabilities for aircraft carriers requires consideration of four factors:

(i)   repair requirements for the ship,

(ii)   forward deployment requirements by the Navy,

(iii)   availability of the shipyards, and

(iv)   capacity of the shipyards.

The limitations associated with shipyard capacity and availability are as follows:

a. Dry docking Capacity And Availability

PUGET has two dry-docks that can handle aircraft carriers. There is one large dry dock for all carriers (nuclear or non-nuclear), and a second slightly smaller one that can handle only non-nuclear carriers. NORVA has one dry dock that can handle either nuclear or non-nuclear carriers. NEWS has two dry docks available which can handle any size carrier. NEWS also has several building docks that are used for carrier construction.

> "Drydocking can be conducted at any time during a docking availability, and normally takes one quarter of the total availability period to complete. By coordinating drydocking schedules, a shipyard may be able to accommodate simultaneous overhauls." [Brown 1998]

b. Repair Man-day Availability

Repair man-days are an important secondary consideration when scheduling maintenance. While there is no limitation on man-day availability, excessive man-day requirements are avoided by staggering depot level maintenance periods to minimize overlaps. Current scheduling practice is to limit the overlap of simultaneous carrier maintenance periods at a given shipyard to 3 months or less in order to avoid man-day shortfalls. [Brown 1998]

Figure VI.5 depicts the overlap of maintenance periods in the same shipyard.

```
CVN 71     5/6/2004          DPIA                  3/17/2005
                             NORVA

CVN 73               9/27/2004      DPIA                8/12/2005
                                    NORVA


        DPIA      : Docking Planned Incremental Availability
        NORVA : Norfolk Naval Shipyard
```
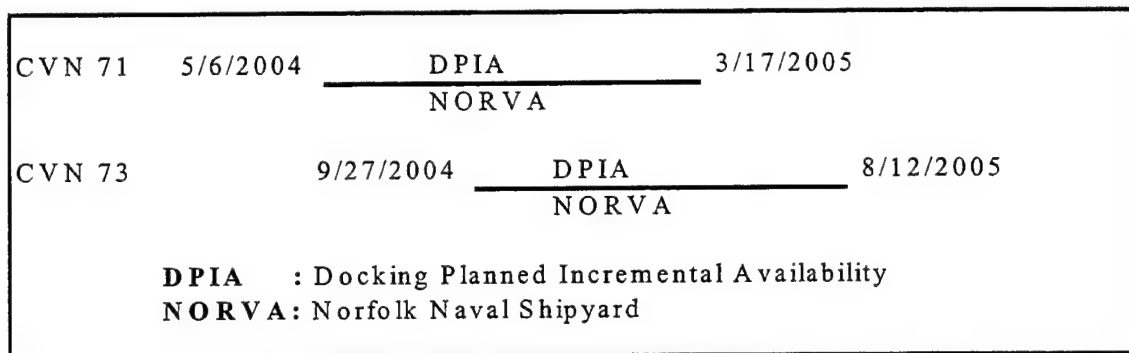
Figure VI.5. Overlap of Two Carrier Maintenance Periods in the Same Shipyard (taken from PERA CV schedule data [OPNAV Report 4710, 1996a]). Scheduled DPIA for CVN 71 begins on May 6, 2004 and ends on March 17, 2005. The DPIA for CVN 73 begins on September 27, 2004 and ends on August 12, 2005. There is an overlap of approximately six months that may adversely affect the man-day availability in the shipyard.

c.   *Refueling Availability*

Nuclear refueling must be conducted at NEWS: "Refueling of nuclear-powered carriers must be conducted at NEWS. Due to the budget constraints, it is preferable to refuel one carrier at a time. However an overlap of at most six months can be accepted." [Brown 1998]

2.   Work-Up Cycle

After depot level maintenance and prior to deployment, all ships are required to execute a Tactical Training Strategy (TTS) that takes place during the period known as the work-up cycle. This work-up cycle ensures that the crew is properly trained and that the ship is ready for deployment.

Conceptually, work-up begins after the completion of maintenance and lasts approximately 11.5 months. However, the record of recent carrier deployments suggests that operational exigencies often curtail pre-deployment work-up.   Work-ups are

scheduled according to the following criteria: When the maintenance period is six months or less, the carrier can deploy fifteen months after the start of maintenance. If the maintenance period is between six and twelve months, then the carrier can deploy nineteen months after the start of maintenance. Finally, if the maintenance is a refueling complex overhaul (RCOH), or the carrier has just been commissioned, it cannot deploy for twelve months after the completion of maintenance or port-shakedown availability. [Brown et al. 1997, pg. 37]

3. Personnel Tempo Of Operations

Navy policy restricts at-sea time in the interest of promising personnel a reasonable amount of time stationed in home port with the families.

> "In order to ensure a balance between the support of national objectives and reasonable operating conditions for Naval personnel, the CNO initiated the Personnel Tempo of Operations (PERSTEMPO) program. The PERSTEMPO program achieves this balance by placing peacetime utilization limitations on all Navy units deployed from their homeport. There are three utilization limitations:
>
> > (1) The maximum length of a deployment cannot exceed six months (180 days).
> >
> > (2) There must be a minimum of a 2-to-1 Turn Around ratio (TAR) between deployments. This means that a carrier must remain home for at least 12 months following a six-month deployment.
> >
> > (3) Over the course of a five-year cycle (three years historical, two years projected), a carrier must spend a minimum of 50% of its time in homeport.
>
> A carrier cannot deploy unless it satisfies these PERSTEMPO restrictions." [OPNAV 1990]

A memorandum from N81 concludes that a TAR of 2.61 to 1 is more reasonable [Brown et al. 1997, pg. 37].

4. Transit Time

Per OPNAV guidance, the transit time between San Diego and the Persian Gulf is 45 days [Brown et al. 1997, pg. 37]. PACFLT carriers from Bremerton or Everett, Washington must transit to San Diego to load the air wing before heading west toward the Persian Gulf. This adds six days to the transit time in both directions. For LANTFLT carriers, the transit time from Norfolk or Mayport to EUCOM is 13 days. However, it takes only eleven days for LANTFLT carriers to return to their homeports.

5. Availability Of LANTFLT Carriers For CENTCOM

LANTFLT carriers can be deployed to CENTCOM to compensate for the loss of coverage due to the longer transit time required for PACFLT carriers to reach CENTCOM. A memorandum from N81 establishes that the LANTFLT carriers should provide 24% of CENTCOM coverage [Brown et al. 1997, pg. 37].

C. SCHEDULE PERIODS

A carrier is in one of the following states during each period:

(i) maintenance,

(ii) work-up,

(iii) *deployable*, or

(iv) *non-deployable*.

When the amount of time between the end of one work-up period and the next maintenance period is at least 180 days, then a deployment is possible. This block of

time is referred to as a deployable period. If the total number of such intervening available days is less than 180, it is a non-deployable period.

Figure VI.6 displays a sample two-year schedule for four carriers (A, B, C, and D). For the purposes of illustration, the time resolution is chosen to be in *months* for each period of the planning horizon.
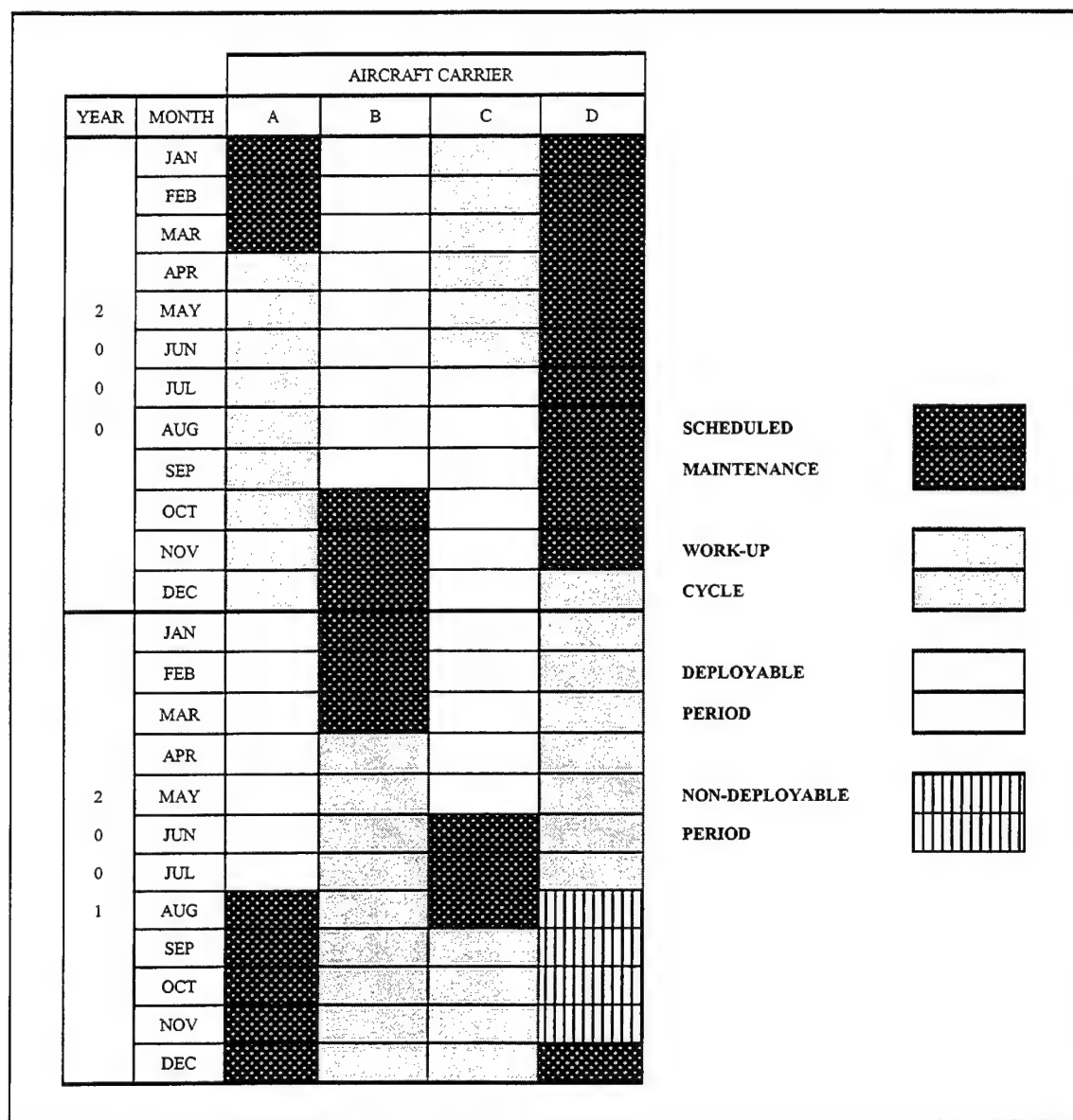
AIRCRAFT CARRIER

YEAR	MONTH	A	B	C	D
2 0 0 0	JAN	▓ Maintenance	Deployable	Work-up	▓ Maintenance
	FEB	▓ Maintenance	Deployable	Work-up	▓ Maintenance
	MAR	▓ Maintenance	Deployable	Work-up	▓ Maintenance
	APR	Work-up	Deployable	Work-up	▓ Maintenance
	MAY	Work-up	Deployable	Work-up	▓ Maintenance
	JUN	Work-up	Deployable	Work-up	▓ Maintenance
	JUL	Work-up	Deployable	Deployable	▓ Maintenance
	AUG	Work-up	Deployable	Deployable	▓ Maintenance
	SEP	Work-up	Deployable	Deployable	▓ Maintenance
	OCT	Work-up	▓ Maintenance	Deployable	▓ Maintenance
	NOV	Work-up	▓ Maintenance	Deployable	▓ Maintenance
	DEC	Deployable	▓ Maintenance	Deployable	Work-up
2 0 0 1	JAN	Deployable	▓ Maintenance	Deployable	Work-up
	FEB	Deployable	▓ Maintenance	Deployable	Work-up
	MAR	Deployable	▓ Maintenance	Deployable	Work-up
	APR	Deployable	Work-up	Deployable	Work-up
	MAY	Deployable	Work-up	Deployable	Work-up
	JUN	Deployable	Work-up	▓ Maintenance	Work-up
	JUL	Deployable	Work-up	▓ Maintenance	Work-up
	AUG	▓ Maintenance	Work-up	▓ Maintenance	Non-deployable
	SEP	▓ Maintenance	Work-up	Deployable	Non-deployable
	OCT	▓ Maintenance	Work-up	Deployable	Non-deployable
	NOV	▓ Maintenance	Work-up	Deployable	Non-deployable
	DEC	▓ Maintenance	Work-up	Deployable	▓ Maintenance

Legend:

SCHEDULED MAINTENANCE	(dark shaded)
WORK-UP CYCLE	(light shaded)
DEPLOYABLE PERIOD	(blank)
NON-DEPLOYABLE PERIOD	(vertical striped)

Figure VI.6. A Sample Two-Year Schedule for Aircraft Carriers. The dark shaded cells represent time in maintenance. Following each maintenance period is a sequence of light shaded cells to indicate the required work-up period. Blank cells represent deployable periods, and vertical striped cells represent non-deployable periods.

117

1. Shifting Maintenance Periods

Maintenance availabilities can be shifted to increase AOR coverage as follows. Figure VI.7 depicts a deployment cycle of an aircraft carrier. If we shift the former maintenance period in Figure VI.7 one month earlier (to the left), then this maintenance period will be completed at the end of the second month. Therefore, the work-up period, and hence the deployable period, will also shift and begin one month earlier. Eventually, the new deployable period will last for 8 months, beginning in month 15 and ending in month 22. In addition, we can also increase the deployable period by one month by shifting the later maintenance period (to the right) so that it begins one month later. Figure VI.8 depicts the effects of such shifting.
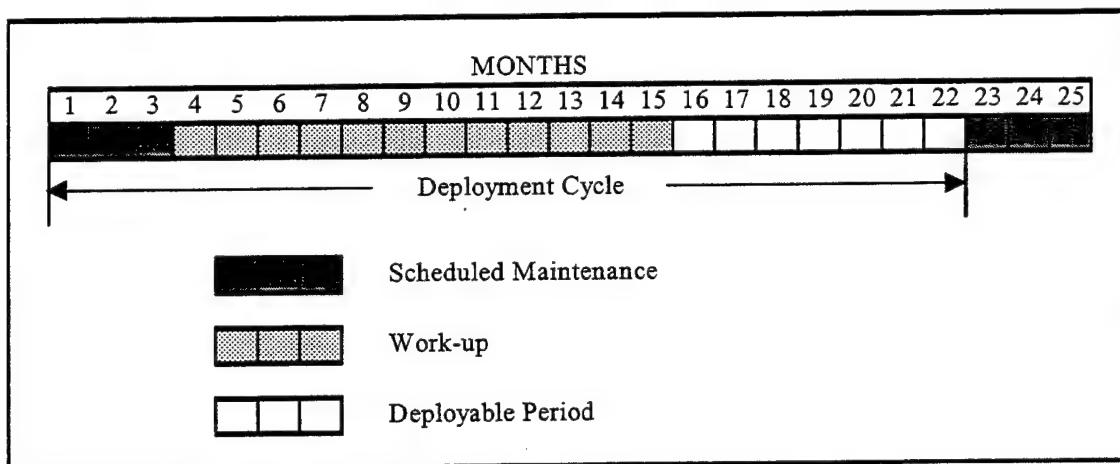

Figure VI.7. A Deployment Cycle for an Aircraft Carrier. The cycle begins with a maintenance period of three months, followed by a work-up period of twelve months, and ends with a deployable period of seven months beginning with month 16 and ending in month 22, after which another maintenance is scheduled.
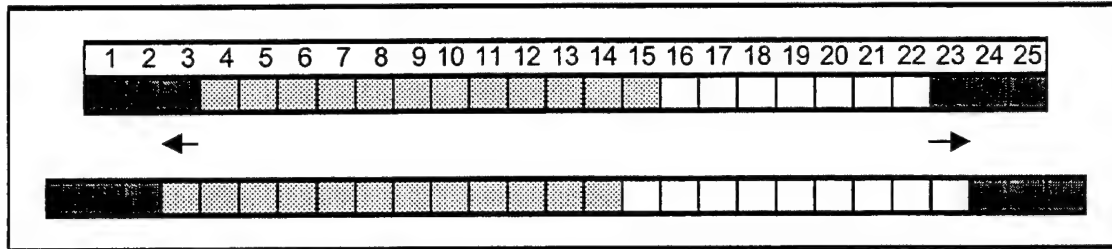
118

Figure VI.8. Shifting of Maintenance Periods. From Figure VI.7, the former (left) maintenance period is shifted one month earlier (to the left), and the second period one month later (to the right) increasing the length of the deployable period by two months.

Shifting maintenance periods may cause an undesirable overlap. This problem will arise if we shift one or both of any two maintenance periods towards each other causing an overlap that exceeds the allowable dry dock, refueling, or man-day availability limits. Potentially overlapping maintenance periods are called *critical maintenance pairs*. The maintenance period that starts earlier is called the *first element of the pair*, and the other maintenance period is called the *second element of the pair*. Figure VI.9 depicts a sample critical maintenance pair.
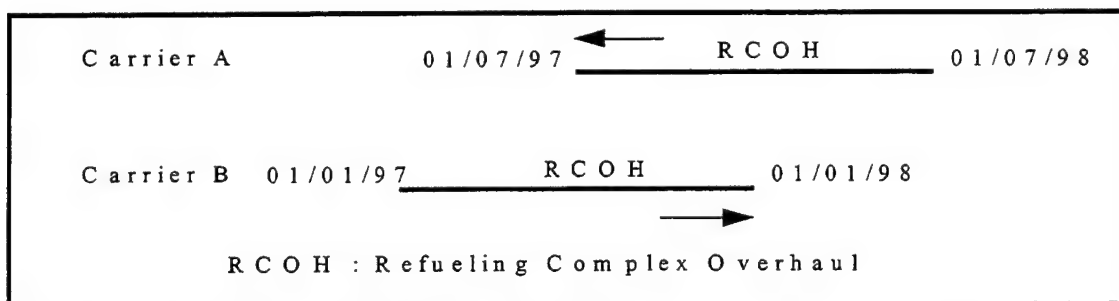

Figure VI.9. A Sample Critical Maintenance Pair. The overlap between two Refueling Complex Overhauls (RCOHs) is six months, which is the maximum allowable limit for refueling nuclear-powered aircraft carriers. If we shift one or both RCOHs towards each other, as shown with the arrows, then the overlap will exceed this limit. However if we shift them in the same direction, the overlap will not change and the refueling constraint will not be violated. The RCOH of Carrier B is the first element of the pair, and the other RCOH is the second element.

119

2. Possible Deployment Schedules In A Deployable Period

a. For PACFLT Carriers

If we assume the carrier in Figure VI.7 to be a PACFLT carrier, then it can be deployed in only one possible way as shown in Figure VI.10.
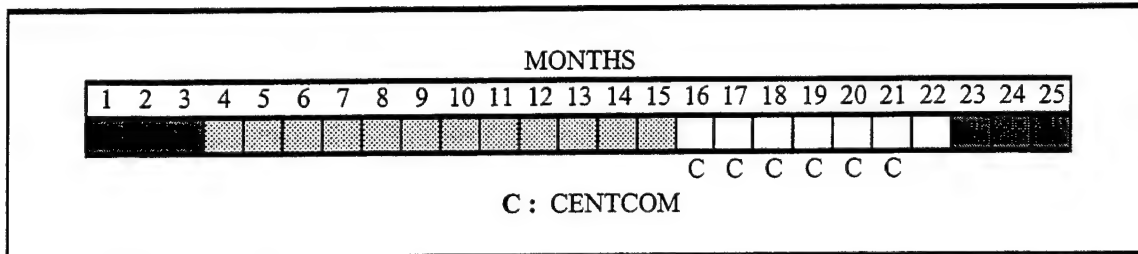

Figure VI.10. From Figure VI.7, a Possible PACFLT Deployment Schedule for a Deployable Period. The carrier can be deployed to CENTCOM from the beginning of the month 16 to the end of the month 21. The carrier is not deployed in month 22, because there should be at least a one-month delay between the end of a deployment period and the start of a maintenance period [Brown et al. 1997, pg. 59].

After maintenance shifts of Figure VI.8, we can deploy this carrier in three alternate ways (Figure VI.11).
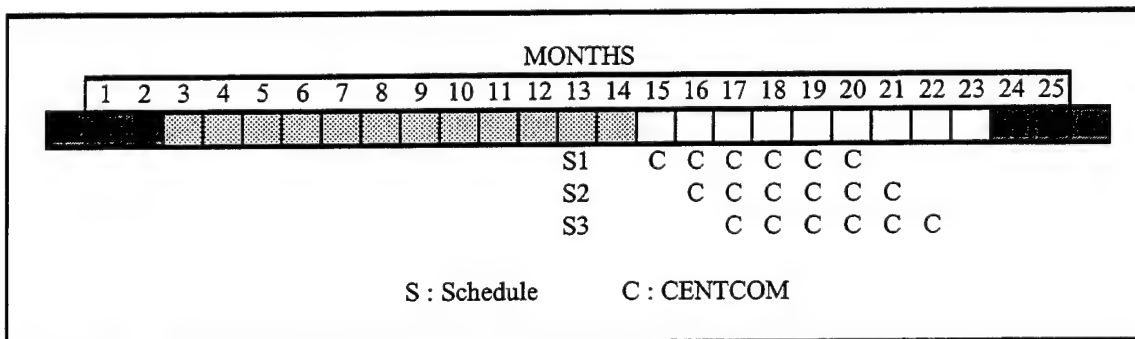

Figure VI.11. Alternate Candidate Schedules for the PACFLT Carrier in the Shifted Deployable Period of Figure VI.8. S1 is an early deployment, S2 is a normal deployment, and S3 is a late deployment.

120

The first deployment schedule (S1) in Figure VI.11 is defined as an *early deployment*. It becomes available by shifting the preceding maintenance period one month earlier. The second schedule (S2) is called a *normal deployment*. This deployment schedule requires no maintenance shifting. The third schedule (S3), a *late deployment*, is obtained by shifting the following maintenance period one month later. For the special case of a seven-month deployable period, obtained by expanding a five-month non-deployable period by two months (one month from each end), the result is called an *early-or-late deployment*.

If we select an early deployment in an optimized solution, then the preceding maintenance period must be shifted to begin one month earlier. If a normal deployment is selected, no change is required. A late deployment requires that the following maintenance period begin one month later. Finally, if an early-or-late deployment is selected, both preceding and following maintenance periods must be shifted away one month.

b. For LANTFLT Carriers

A LANTFLT carrier can deploy to EUCOM for twenty-one weeks, or alternatively, to EUCOM for 15 weeks and CENTCOM for an additional six weeks. Figure VI.12 depicts possible schedules for a sample deployment cycle of a LANTFLT carrier, and also shows the scheme of EUCOM and CENTCOM coverage combinations. (For purposes of illustration using monthly increments, it is assumed in the figure that CENTCOM coverage is two months instead of six weeks.)
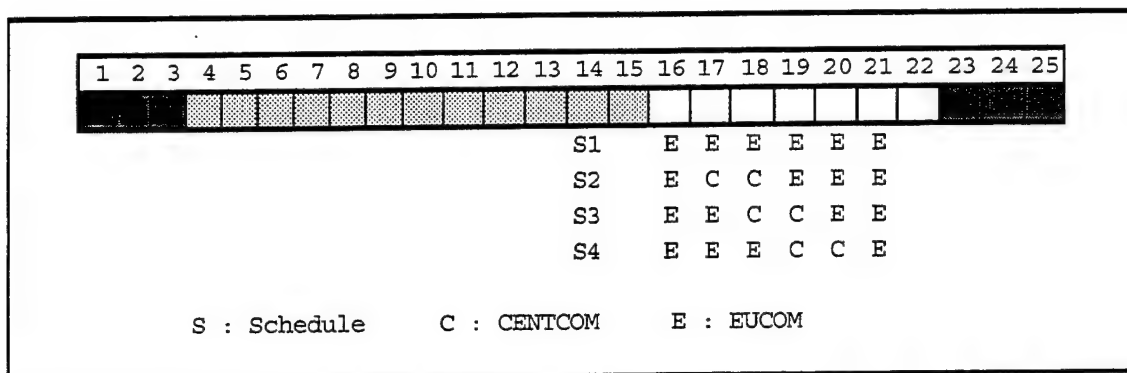
121

```
     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

                                         S1        E   E   E   E   E   E
                                         S2        E   C   C   E   E   E
                                         S3        E   E   C   C   E   E
                                         S4        E   E   E   C   C   E

          S  :  Schedule        C  :  CENTCOM      E  :  EUCOM
```

Figure VI.12. Possible Schedules for a Sample Deployment Cycle of a LANTFLT Carrier. The carrier can be deployed either for six months in EUCOM, or for a combination of two months of CENTCOM deployment and four months of EUCOM deployment.

D.   TWO-COMMODITY NETWORK FLOW PROBLEM WITH SIDE CONSTRAINTS

1.   Optimization Model Generation

Possible deployment schedules for each carrier and deployment cycle are generated according to the criteria stated in the previous sections. After generating all possible deployment schedules, the next step is to find all feasible combinations of these schedules. The coverage gaps between each pair of schedules should not exceed a specified length of time that is referred to as *max-gap* [Schauppner 1996, pg. 15].

When a feasible combination of schedules is sorted chronologically, every pair of successive schedules must belong to different deployable periods. If two successive schedules i and j belong to carriers A and B, respectively, in the same fleet, then carrier A should depart its homeport before carrier B. This ensures that the two carriers do not cover an AOR for exactly the same period. Two deployment schedules are said to be *compatible* when they satisfy these conditions.

122

Table VI.2, Figure VI.13, and Figure VI.14 illustrate compatible and incompatible pairs of deployment schedules. Table VI.2 shows a sample two-year schedule for four aircraft carriers. The first two are PACFLT carriers and the other two are from LANTFLT. The deployable periods of the carriers in Table VI.2 are already increased by shifting the associated maintenance and work-up periods. Figure VI.12 displays coverage gaps and coverage *overlaps* (the length of time in which more than one carrier is in an AOR) that will occur in CENTCOM for each pair of compatible deployment schedules. Figure VI.15 displays coverage gaps for EUCOM. For purposes of illustration only, the transit times to and from CENTCOM are assumed here to be 30 days each for PACFLT carriers. For LANTFLT carriers, the transit times to and from EUCOM are assumed here to be 15 days. The transit times between CENTCOM and EUCOM are not taken into account here.

MONTHS	AE1	AN2	AN3	AL4	BE1	BL2	CE1	CE2	CE3	CE4	CL5	CL6	CL7	CL8	DEL1	DEL2	DEL3	DEL4
1																		
2															E	E	E	E
3	C														E	C	E	E
4	C	C													E	C	C	E
5	C	C	C												E	E	C	C
6	C	C	C	C											E	E	E	C
7	C	C	C	C											E	E	E	E
8	C	C	C	C														
9		C	C	C														
10			C	C														
11						C		E	E	E	E							
12						C		E	C	E	E	E	E	E	E			
13								E	C	C	E	E	C	E	E			
14					C			E	E	C	C	E	C	C	E			
15					C	C		E	E	E	C	E	E	C	C			
16					C	C		E	E	E	E	E	E	E	C			
17					C	C						E	E	E	E			
18					C	C												
19					C	C												
20						C												
21																		
22																		
23																		
24																		

Table VI.2. A Sample Two-year Schedule for Four Aircraft Carriers. Each row represents a month, and each column represents a deployment schedule. The first column is labeled AE1. A represents the carrier, E indicates that the schedule is an early deployment, and 1 is the schedule number for the carrier in this deployable period. A column label with second character N, represents a normal deployment, L means a late deployment, and EL an early-or-late deployment.

	AE1	AN2	AN3	AL4	BE1	BL2	CE1	CE2	CE3	CE4	CL5	CL6	CL7	CL8
AE1					7	8		4	5	6		5	6	7
AN2								3	4	5		4	5	6
AN3								2	3	4		3	4	5
AL4								1	2	3		2	3	4
BE1														
BL2														
CE1														
CE2					1	2								
CE3					0	1								
CE4					-1	0								
CL5														
CL6					0	1								
CL7					-1	0								
CL8					-2	-1								
DEL1														
DEL2	-1	0	1	2	10	11		7	8	9		9	10	11
DEL3	-2	-1	0	1	9	10		6	7	8		8	9	10
DEL4	-3	-2	-1	0	8	9		5	6	7		7	8	9

Figure VI.13. Coverage Gaps and Overlaps that Accrue in CENTCOM for Each Pair of Compatible Deployment Schedules. Each row indicates the first carrier in a pair, and each column the second. For example, cell (AE1, BE1), has a value of 7: if we first deploy carrier A with AE1, and then deploy carrier B with BE1, then there will be a coverage gap of seven months in CENTCOM, starting from the beginning of AE1 and ending at the end of BE1. A negative number indicates overlap periods. For example, cell (DEL2, AE1) shows that carriers D and A will cover CENTCOM together for one month. Blank cells represent incompatible pairs or pairs for which one or both schedules do not provide any coverage of CENTCOM.

124

First we calculate overlap values. Then, for the purposes of avoiding undesired amounts of overlap, the values that are longer than a specific maximum overlap value are eliminated. The remaining overlap values are assigned to zero, in order to restate them in terms of coverage gaps.

	CE1	CE2	CE3	CE4	CL5	CL6	CL7	CL8
CE1								
CE2								
CE3								
CE4								
CL5								
CL6								
CL7								
CL8								
DEL1	4	4	4	4	5	5	5	5
DEL2	6	6	6	6	7	7	7	7
DEL3	6	6	6	6	7	7	7	7
DEL4	6	6	6	6	7	7	7	7

Figure VI.14. EUCOM Coverage Gaps for Each Pair of Compatible Deployment Schedules. Unlike the situation shown in Figure VI.13, there are no overlaps here.

Note that Figures VI.13 and VI.14 are node-node adjacency matrices of a network (e.g., [Ahuja et al. 1993]). Figure VI.15 represents the network underlying Figure VI.14. Nodes S1 and T1 are added to represent the starting and termination of the planning horizon for EUCOM coverage. The other nodes correspond to deployment schedules for carriers. Costs associated with arcs originating from node S1 and terminating at schedules DEL1 to DEL4 correspond to the coverage gap (in months) from the beginning of the planning horizon to the start of EUCOM coverage by carrier D. To simplify Figure VI.15, arcs from S1 to nodes CE1 through CL8 are not shown. Similarly, costs associated with arcs from schedules CE1 through CL8 to node T1 correspond to the coverage gap evident from the end of coverage by carrier E to the end of the planning

125

horizon. A network corresponding to Figure VI.13 is similarly constructed with nodes S2 and T2 added to represent the start and termination of the planning horizon for CENTCOM.
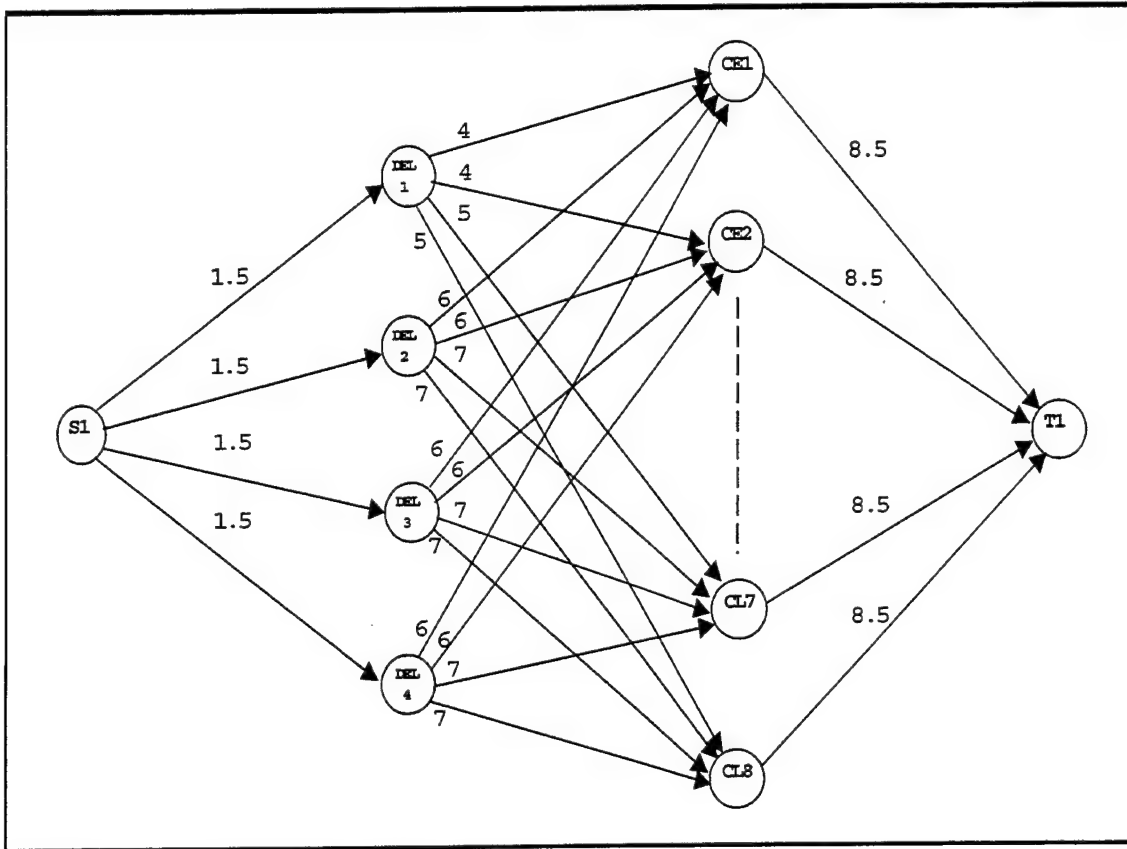

Figure VI.15. Network Depicting Possible Sequencing of Deployment Schedules for EUCOM. S1 is the start node, and T1 the termination node of the network indicating the beginning and end of the planning horizon, respectively. Any other node in the network represents a deployment schedule. Each arc length corresponds to the coverage gap between the two associated deployment schedules. Nodes CE3, CE4, CL5, and CL6 exist, but are not displayed in the figure.

For each of the two networks (one for CENTCOM and one for EUCOM) described above, we can derive feasible paths beginning with S1 or S2, visiting at most one node (or schedule) in each deployable period, and ending with T1 or T2. Figure

126

VI.16 depicts two sample feasible paths, one derived from the CENTCOM network, and the other from the EUCOM network. Because a LANTFLT carrier can cover both CENTCOM and EUCOM in the same deployment schedule, a feasible path from the CENTCOM network may have common nodes with another feasible path from the EUCOM network.
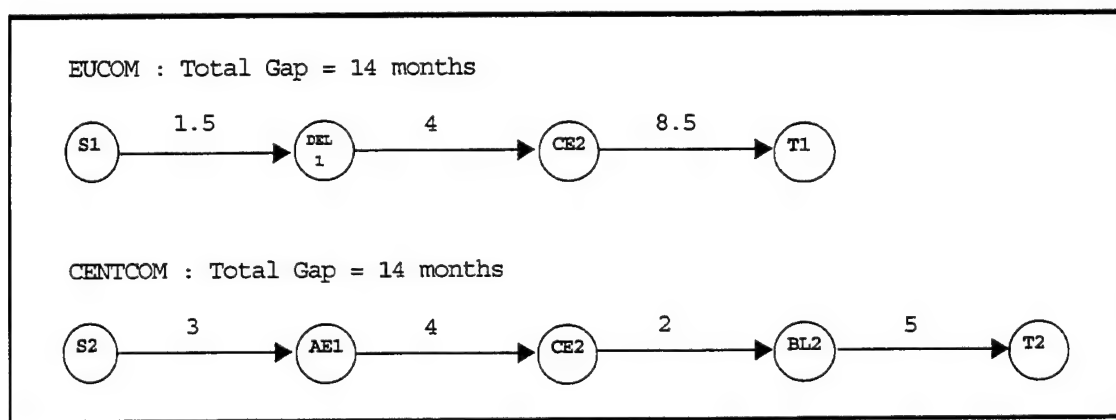

Figure VI.16. Two Sample Feasible Paths: One Derived from the EUCOM Network, and the Other from the CENTCOM Network. Node CE2 is common to both paths, meaning that LANTFLT carrier covers both EUCOM and CENTCOM in the same schedule. Each arc length corresponds to the coverage gap, in months, between the two associated deployment schedules.

At this point, the carrier deployment problem is reduced to finding two paths that satisfy the conditions discussed above, and that yield a minimum gap. This problem can be formulated as a two-commodity network flow problem with side constraints.

2.   Model Formulation

Using the two-commodity network model with side constraints, the mathematical formulation for the long-term synchronous aircraft carrier deployment and major maintenance scheduling problem is provided below:

Indices:

a AORs (EUCOM and CENTCOM)

c carriers

d deployable periods $(1, 2,..., D{-}1, D)$

i, j nodes in the networks representing the schedules (Nodes S^a and T^a represent the beginning and the end of the planning horizon, respectively)

s dry docking or refueling shipyards (Newport News, Puget Sound, and Norfolk)

p critical maintenance period pairs (i.e., maintenance periods that will overlap more than the allowable dry docking, refueling, or man-day availability limits when shifted towards each other).

Index Maps:

Φ^a = {c: carrier c can cover AOR a}

Ω_c^d = {i: schedule i belongs to deployable period d of carrier c}

Θ_c^d = {i: schedule i belongs to early deployments of period d of carrier c}

Λ_c^d = {i: schedule i belongs to late deployments of period d of carrier c}

Γ = {i: schedule i covers both AORs}

Δ^c = {d: period d of carrier c is obtained as deployable by shifting maintenance periods}

E_s^p = {i: schedule i belongs to early deployments of the deployable period right

after the second element of critical maintenance period pair p (i.e., the

maintenance period which starts later) for shipyard s }

L_s^p = {i: schedule i belongs to late deployments of the deployable period right

before the first element of critical maintenance period pair p (i.e. the

maintenance period which starts earlier) for shipyard s }

Data:

GAP_{ij}^a gap length in AOR a if node j follows node i in a path

$EXIST_{ij}^a$ equals 1 if there is an arc from node i to node j for AOR a

$WEIGHT^a$ weight for coverage gap in AOR a (e.g., a CENTCOM weight of

3, and a EUCOM weight of 1)

$CONST$ sufficiently small number to penalize the early or late deployments

(units of months)

(Binary) Decision Variables

x_{ij}^a equals 1 if arc (i,j) belongs to the path from S^a to T^a and 0 otherwise

Formulation

$$\text{minimize} \quad \sum_a WEIGHT^a \sum_{\{(i,j):EXIST_{ij}^a=1\}} GAP_{ij}^a x_{ij}^a \quad + \sum_{\left\{\substack{(a,c,d,i,j):c\in\Phi^a, \\ i\in(\Theta_c^d or \Lambda_c^d), EXIST_{ij}^a=1}\right\}} CONST\, x_{ij}^a \qquad (VI.1.a)$$

s.t.

$$\sum_{\{j:EXIST_{ji}^a=1\}} x_{ji}^a - \sum_{\{j:EXIST_{ij}^a=1\}} x_{ij}^a = \begin{cases} -1 \text{ if } i=S^a \\ \;\;\,1 \text{ if } i=T^a \qquad \forall\, a,\, i \\ \;\;\;\,0 \text{ otherwise} \end{cases} \qquad (VI.1.b)$$

$$\sum_{\{i\in\Omega_c^d\}}\sum_{\{j:EXIST_{ij}^a=1\}} x_{ij}^a \le 1 \qquad\qquad \forall\, a,\, d,\, \text{and } c\in\Phi^a \qquad (VI.1.c)$$

$$\sum_{\{j:EXIST_{ij}^{EUCOM}=1\}} x_{ij}^{EUCOM} - \sum_{\{j:EXIST_{ij}^{CENTCOM}=1\}} x_{ij}^{CENTCOM} \ge 0 \quad \forall\, i\in\Gamma \qquad (VI.1.d)$$

$$\sum_{\{(i,j):i\in L_s^p, EXIST_{ij}^a=1\}} x_{ij}^a + \sum_{\{(i,j):i\in E_s^p, EXIST_{ij}^a=1\}} x_{ij}^a \le 1 \qquad \forall\, a,\, s,\, \text{and } p \qquad (VI.1.e)$$

$$\sum_{\{(i,j):i\in\Lambda_c^d, EXIST_{ij}^a=1\}} x_{ij}^a + \sum_{\{(i,j):i\in\Theta_c^d, EXIST_{ij}^a=1\}} x_{ij}^a \le 1 \qquad \forall\, a,\, d\in\{1,\ldots,D-1\},\, \text{and } c\in\Phi^a \qquad (VI.1.f)$$

$$\sum_{\{(i,j):i\in\Lambda_c^{d-1}, EXIST_{ij}^a=1\}} x_{ij}^a + \sum_{\{(i,j):i\in\Lambda_c^d, EXIST_{ij}^a=1\}} x_{ij}^a \le 1 \qquad \forall\, a,\, d\in(\Delta^c \text{ and } \{2,\ldots,D\}),\, \text{and } c\in\Phi^a \quad (VI.1.g)$$

$$\sum_{\{(i,j):i\in\Theta_c^d, EXIST_{ij}^a=1\}} x_{ij}^a + \sum_{\{(i,j):i\in\Theta_c^{d+1}, EXIST_{ij}^a=1\}} x_{ij}^a \le 1 \quad \forall\, a,\, d\in(\Delta^c \text{ and } \{1,\ldots,D-1\}),\, \text{and } c\in\Phi^a \quad (VI.1.h)$$

In the above formulation, the objective is to minimize the weighted coverage gaps in CENTCOM and EUCOM with a very small penalty assessed for shifted maintenance schedules. The optimization model employs a weighting scheme that accentuates CENTCOM coverage more than EUCOM so as to generate deployment schedules in which LANTFLT carriers provide approximately 24% of CENTCOM coverage.

Constraint (VI.1.b), a flow balance constraint, ensures that there is a path from S^a to T^a. Constraint (VI.1.c) ensures that at most one schedule is selected from each deployable period. Constraint (VI.1.d) ensures that the same schedule is selected, if it covers both AORs. Constraint (VI.1.e) ensures that the dry docking or refueling capacity of shipyard s is not exceeded. Constraint (VI.1.f) ensures that a maintenance period is shifted in one direction only (i.e., a maintenance period cannot start one month late and simultaneously end one month early).

Constraint (VI.1.g) ensures that a carrier cannot be deployed in deployable period d that was obtained from a non-deployable period, and at the same time deployed in a late deployment schedule of period $(d-1)$. If we deploy a carrier in a late deployment schedule of period $(d-1)$, then this will preclude deployment in period d. Therefore, a late deployment in $(d-1)$ is mutually exclusive with any deployment in d. Constraint (VI.1.h) ensures that a carrier cannot simultaneously be deployed in deployable period d that was obtained from a non-deployable period, and in an early deployment schedule of period $(d+1)$.

E.   NEW FORMULATION: DECOUPLING DEPLOYABLE PERIODS

The new formulation introduced in this section is conceptually similar to the classical set partitioning formulation for ship scheduling problems that is presented in Chapter I, Section A.3. However, unlike the classical approach, an alternate schedule does not involve the period-by-period status of the carrier for the entire planning horizon. Each carrier's planning horizon is decoupled into its deployable periods, and possible deployment schedules are generated for each of these deployable periods.

The algebraic formulation of the new formulation is as follows:

Indices:

c carriers

a AORs (CENTCOM, EUCOM)

d deployable periods $(1, 2,..., D{-}1, D)$

t periods $(1,...,T)$

s dry docking or refueling shipyards (Newport News, Puget Sound, and

          Norfolk).

p critical maintenance period pairs (i.e., maintenance periods that

          will overlap more than the allowable dry docking, refueling, or man-day

          availability limits when shifted towards each other).

$j \in J_d^c$ set of *possible schedules* for deployable period d of carrier c (i.e.,

          schedules that satisfy the operations and maintenance constraints, and

          provide the period-by-period status of this carrier for deployable period d).

132

Index Maps:

Θ_d^c = {j: schedule j belongs to early deployments of period d of carrier c}

Λ_d^c = {j: schedule j belongs to late deployments of period d of carrier c}

Δ^c = {d: period d of carrier c is obtained as deployable by shifting

maintenance periods}

E_s^p = {j: schedule j belongs to early deployments of the deployable period right

after the second element of critical maintenance period pair p (i.e., the

maintenance period which starts later) for shipyard s }

L_s^p = {j: schedule j belongs to late deployments of the deployable period right

before the first element of critical maintenance period pair p (i.e. the

maintenance period which starts earlier) for shipyard s }

Data:

A_{dtj}^{ac} equals 1 if schedule $j \in J_d^c$ covers AOR a in period t, 0 otherwise

$WEIGHT^a$ weight of coverage in AOR a

$MAXGAP$ maximum allowable number of consecutive uncovered periods in an

AOR

Decision Variables (Binary)

x_j equals 1 if schedule j is selected, 0 otherwise

$uncovered_t^a$ equals 1 if AOR a is not covered in period t, 0 otherwise

Formulation

$$\text{minimize} \quad \sum_{a,j} WEIGHT^a uncovered_j^a \qquad\qquad\qquad\qquad \text{(VI.2.a)}$$

s.t.

$$\sum_{j \in J_d^c} x_j = 1 \qquad\qquad\qquad \forall\ c, d \qquad\qquad\qquad \text{(VI.2.b)}$$

$$\sum_{c,d,j} A_{dij}^{ac} x_j + uncovered_t^a \geq 1 \qquad\qquad \forall\ a, t \qquad\qquad \text{(VI.2.c)}$$

$$\sum_{t'=t}^{t+(MAXGAP-1)} uncovered_{t'}^a \leq (MAXGAP-1) \qquad \forall\ a,\ t \in \{1,...,T-(MAXGAP-1)\} \quad \text{(VI.2.d)}$$

$$\sum_{j \in L_s^p} x_j + \sum_{j \in E_s^p} x_j \leq 1 \qquad\qquad \forall\ s, \text{and } p \qquad\qquad \text{(VI.2.e)}$$

$$\sum_{j \in \Lambda_d^c} x_j + \sum_{j \in \Theta_d^c} x_j \leq 1 \qquad\qquad \forall\ c, d \in \{1,...,D-1\} \qquad \text{(VI.2.f)}$$

$$\sum_{j \in \Lambda_{d-1}^c} x_j + \sum_{j \in \Lambda_d^c} x_j \leq 1 \qquad\qquad \forall\ c, d \in (\Delta^c \text{ and } \{2,...,D\}) \qquad \text{(VI.2.g)}$$

$$\sum_{j \in \Theta_d^c} x_j + \sum_{j \in \Theta_{d+1}^c} x_j \leq 1 \qquad\qquad \forall\ c, d \in (\Delta^c \text{ and } \{1,...,D-1\}) \qquad \text{(VI.2.h)}$$

In the above formulation, the objective is to minimize the uncovered periods in each AOR. Partition constraints (VI.2.b) ensure that exactly one schedule is selected for each deployable period. Constraints (VI.2.c) express that each AOR should be covered in each period. Because covering all AORs is not possible with the current carrier force,

134

this constraint is elasticized using a penalized elastic variable for each uncovered period. Packing constraints (VI.2.d) ensure that the uncovered period for each AOR is no more than the maximum allowable number of gap periods (*MAXGAP*).

Constraints (VI.2.e) ensure that the dry docking or refueling capacity of shipyard s is not exceeded. Constraints (VI.2.f) ensure that a maintenance period is shifted in one direction only (i.e., a maintenance period cannot start one month late and simultaneously end one month early). Constraints (VI.2.g) ensure that a carrier cannot be deployed in deployable period d that was obtained from a non-deployable period, and at the same time deployed in a late deployment schedule of period $(d-1)$. Constraints (VI.2.h) ensure that a carrier cannot simultaneously be deployed in deployable period d that was obtained from a non-deployable period, and in an early deployment schedule of period $(d+1)$.

Notice that constraints (VI.2.b), (VI.2.d), (VI.2.f), (VI.2.g) and (VI.2.h) have exactly one segment of consecutive ones in each row, while constraints (VI.2.e) have two. Each constraint of (VI.2.c) has segments of consecutive ones in its rows, as well as in its columns. In practice, the number of segments of ones in constraints (VI.2.c) are no more than five in each row, representing that at any period of time at most four carriers are available for coverage of each AOR (plus one segment for the elastic variable *uncovered*). Additionally, the number of consecutive ones segments is no more than three in each (VI.2.c) column. Because a LANTFLT carrier may cover more than one AOR for each deployable period, there are three segments in each associated column.

F.   COMPARISON OF FORMULATIONS

Table VI.3 presents the problem size comparison of the two-commodity network model with side constraints, and the new decoupled set partitioning formulation introduced in the previous section.

Scenario and Time Fidelity	Two-Commodity Network with Side Constraints		New Formulation: Decoupling Deployable Periods	
	Number of Constraints	Number of Columns	Number of Constraints	Number of Columns
Shifted Maintenance Four-week Periods	(VI.1.b) 1,600 (VI.1.c) 50 (VI.1.d) 437 (VI.1.e) 5 (VI.1.f) 50 (VI.1.g) 9 (VI.1.h) 9		(VI.2.b) 50 (VI.2.c) 1,046 (VI.2.d) 1,028 (VI.2.e) 5 (VI.2.f) 50 (VI.2.g) 9 (VI.2.h) 9	
Total	2,160	45,675	2,197	1,918
Shifted Maintenance One-week Periods	(VI.1.b) 14,979 (VI.1.c) 50 (VI.1.d) 4,074 (VI.1.e) 5 (VI.1.f) 50 (VI.1.g) 9 (VI.1.h) 9		(VI.2.b) 50 (VI.2.c) 1,046 (VI.2.d) 1,028 (VI.2.e) 5 (VI.2.f) 50 (VI.2.g) 9 (VI.2.h) 9	
Total	19,176	2,282,015	2,197	9,210
Fixed Maintenance One-week Periods	(VI.1.b) 4,739 (VI.1.c) 41 (VI.1.d) 1,289		(VI.2.b) 41 (VI.2.c) 1,046 (VI.2.d) 1,028	
Total	6,078	258,490	2,115	3,629

Table VI.3. Problem Size Comparison of the Two-Commodity Network Model with Side Constraints and the New Formulation: Decoupling Deployable Periods. The two formulations are compared in the context of three scenarios using the ten-year PERA CV schedule data taken from OPNAV Report 4710 [1996a] for twelve carriers. In the shifted maintenance scenarios, the optimization model is allowed to advance or delay the maintenance periods at most four weeks to obtain better coverage percentages. By contrast, in the fixed maintenance scenario, coverage is maximized around fixed maintenance availabilities. Four-week (or one-week) periods indicate that a carrier may be scheduled to deploy at the beginning of every four-week (or one-week) interval. The number of rows specified for each constraint type is provided next to the associated equation numbers.

136

To date, the two-commodity network with side constraints model has been implemented for the aircraft carrier scheduling problem using a time resolution of four weeks. That is, a carrier may be scheduled to deploy at the beginning of every four-week interval. The reason for restricting with a four-week time resolution is to render the scheduling problem computationally tractable. However, our computational results on the PERA CV scheduling data for twelve carriers and fixed maintenance periods show that if the time resolution is defined in weeks (a relaxation), then the coverage percentage in CENTCOM AOR **increases 1.88% or ten carrier weeks**. Considering the cost of carrier operations and the desire to maximize coverage in the AORs, such a percentage increase is significant.

Table VI.3 shows that if the time resolution of the aircraft carrier scheduling problem is defined in weeks, the two-commodity network model with side constraints becomes very difficult to solve for even the most basic fixed maintenance scenario. However, the new decoupled set partitioning formulation generates reasonable problem sizes that can be solved efficiently in all scenarios.

Another disadvantage of the two-commodity network model with side constraints is the restriction that the CENTCOM coverage by LANTFLT carriers lasts six weeks. This assumption has been made to reduce the number of possible schedules, and to make the scheduling problem solvable. However, using the new decoupled set partitioning formulation, the number of weeks a LANTFLT carrier should cover CENTCOM can be determined by the optimization. This flexibility will increase the coverage in both AORs, and still generate a reasonable problem size.

137

Finally, we note that the LP relaxation lower bound obtained using the classical set partitioning model is stronger than the respective LP relaxation lower bound obtained using the two-commodity network model with side constraints, or the new decoupled set partitioning formulation. When we relax the respective binary requirements in the two-commodity network model with side constraints or in the new decoupled set partitioning formulation, the mutually exclusive packing constraints (VI.1.e) through (VI.1.h), and (VI.2.e) through (VI.2.h) admit fractional (non-binary) alternate schedules with non-uniform values adding to at most one. Although the decoupled set partition yields a weaker LP relaxation lower bound, it has proven effective for solving the carrier scheduling problem because, even over a long planning horizon, the number of alternate schedules remains modest.

VII. CONCLUSIONS, CONTRIBUTIONS, AND RECOMMENDED FUTURE RESEARCH

Set partitioning problems arise in many applications of optimization-based decision support. Perhaps the best-known of these is airline crew scheduling, where each binary decision (syn. "variable," or "column") represents whether or not to assign a crew to a duty period consisting of a set of (takeoff-to-landing) flights, and each constraint (row) is a flight that must have a crew assigned. These models are deceptively easy to state: find a set of columns that make exactly one (crew) assignment to each row. But, these problems are quite difficult to solve. Even the continuous, linear programming relaxation is not necessarily easy to solve, and even when it has been solved, there are frequently many variables in the solution with fractional values, so that enumeration with branch-and-bound or branch-and-cut becomes necessary.

Just getting a feasible solution for a set partition, let alone an optimal or near-optimal one, is hard. In fact, even having a bound on such a solution can be a valuable clue about how to actually solve it.

In this dissertation we have isolated a special structure exhibited by set partitions that we call "consecutive ones." A consecutive ones column in an airline crew scheduling problem assigns a consecutive set of flights, and a consecutive ones row is associated with a flight that can be assigned to a consecutive set of candidate columns. Set partitions with consecutive ones columns can be reformulated into an equivalent minimum-cost pure network flow problem, and thus solved easily. Consecutive ones

139

rows also yield an easy reformulation that is the dual of a minimum-cost pure network flow problem.

However, finding a set partition that has all consecutive ones rows or columns is a lucky and rare event. Predominately, we find columns or rows composed of more than one consecutive ones segment. If the number of consecutive ones segments per column or row is small, we call this "near-consecutive ones."

We have looked for, and found, intrinsic near-consecutive ones rows and/or columns in many existing set partitions. We have shown how to reorder columns and/or rows to minimize the respective number of consecutive ones segments in rows and/or columns. We have also formulated set partitions, in particular a long-term US Navy aircraft carrier deployment scheduling model, with the aim of minimizing the number of consecutive ones segments in its rows and/or columns.

Having found near-consecutive ones columns, and perhaps having reduced the number of segments in them, we show how to algebraically reformulate, or "column split" the set partition into two sets of rows with some additional columns such that one set of rows has only consecutive ones columns. We show how each of these row sets is, individually, equivalent to a pure network flow problem, so each is easy to solve quickly. However, directly solving both sets simultaneously is not easy, so we retain the larger set of network constraints and use Lagrangian relaxation with subgradient optimization to attempt to satisfy the smaller set of network constraints by iterative means. This use of column splitting has been suggested before for solving some special problems, such as the two-duty period scheduling problem [Marsten and Shepardson 1980], but never

140

reported for general set partitions. Our results show that it is important not to have too many consecutive ones column segments, or it becomes very difficult to solve the Lagrangian relaxation in a reasonable amount of time. Even in cases where a set partition cannot be solved this way, the solution can be bounded, however.

We also find near-consecutive ones rows, and perhaps reduce the number of segments in these, and show how to use this perspective to reformulate the set partition into two sets of columns with some additional rows. One set of columns has only consecutive ones rows. We show how each of these column sets is, individually, equivalent to a pure network flow problem. We also show that if the variables in one set are fixed, the remaining set conditioned by this can be solved easily. We use this to develop the row split Benders decomposition method to solve set partitions. The results show that is important not to have too many consecutive ones row segments, or the number of Benders iterations required for convergence grows alarmingly. Even in cases where a set partition cannot be solved with row split Benders decomposition, the solution can be bounded.

We do find cases where the column split and/or row split methods solve the set partition faster than conventional means. We conjecture that, given the difficulty of solving set partitions, there exist instances that these new techniques will solve that would defy conventional means.

It is well known that linear programs and mixed integer linear programs (and set partitions) usually exhibit redundant features that can be easily detected and exploited by relaxing constraints and fixing variables. For instance, if a constraint restricts a set of

141

binary variables to sum to one, and if another constraint restricts that same set of variables and another set of binary variables to also sum to one, then the latter set of variables can all be set to zero, and the latter constraint can be relaxed. These detection features, called "presolve," "prereduce," or "preprocessing," are standard features in commercial optimization software packages and are well known to improve solution efficiency, sometimes dramatically.

There are also well known reductions applicable to pure network flow problems. Finding that set partition features can be reformulated to equivalent network flow features suggests that for every reduction applicable to a set partition, there might be some equivalent reduction in the equivalent pure network, and vice versa. We establish equivalence between pairs of set partition and network reductions, and discover that some that have been reported to be distinct are not. We also discover a new set partition reduction from a known network reduction.

A. CONTRIBUTIONS

1. Column Splitting

We have investigated Lagrangian relaxation and subgradient optimization with the column split reformulation for various problems including: (i) an instance of an aircraft carrier problem, (ii) sample data from two-, three-, and four-duty period problems, and (iii) a subset of real-world airline crew scheduling problems that exhibits a reasonable degree of consecutive ones structure in the columns. During the implementation of subgradient optimization, some of the recent improvements reported by Caprara et al. [1999] are incorporated.

For airline crew scheduling problems that do not have a significant consecutive ones structure, the convergence of Lagrangian relaxation is unsatisfactory. However, for instances of two-, three-, and four-duty scheduling scenarios, the Lagrangian relaxation yields a respectable lower bound faster than the LP relaxation of the monolithic SPP. But, obtaining a lower bound within 1% of the LP relaxation lower bound using Lagrangian relaxation usually requires an excessive number of iterations and long solution times. As the number of segments of consecutive ones increases, the convergence speed of Lagrangian relaxation deteriorates.

We have also investigated an exterior penalty method on the column split reformulation to find a lower bound that is equal to the LP relaxation of SPP. Our penalty subproblem is a convex quadratic network flow problem. We have implemented our penalty method using the CPLEX 6.6 [ILOG 2000] quadratic programming solver, which is not specially designed for solving quadratic network flow problems.

For certain problems with near-consecutive ones, our exterior penalty method yields lower bound values faster than the conventional means, including the simplex, dual simplex, or barrier methods. A specialized quadratic network solver (e.g., that reported by Dembo [1987] and Hearn et al. [1987] but not available from those authors) would presumably improve solution times further.

By reordering the rows, we may reduce the number of consecutive ones segments in the columns of an SPP: using a trivial row reordering heuristic on a subset of airline crew scheduling problems, we obtain a 32% average decrease in the number of consecutive ones segments. Furthermore, after reordering the rows, the solution times

143

and the lower bound values obtained using the Lagrangian relaxation and the penalty method improve significantly. For one of the sample problems, we obtain a better solution time using the exterior penalty method than solving the original LP relaxation using the simplex method.

2. Row Split Benders Decomposition

We have introduced a new algorithm to solve certain binary programming problems (e.g. set partitions, their cousins set packs and set covers, and instances of these with additional side constraints) whose rows contain segments of consecutive ones. We reformulate using a row splitting technique that isolates a pure network enabling application of Benders decomposition.

For some problems with near-consecutive ones in rows (e.g., the aircraft carrier scheduling problem), the row split Benders decomposition yields faster solution times. We conjecture that row split Benders decomposition will render some previously unsolvable real-world applications tractable.

Further, we have investigated the impact of column reordering on the performance of the row split Benders decomposition.

We have also tested the performance of the row split Benders decomposition in generating improved feasible solutions for an SP, SC or SPP.

By reordering the columns of a subset of airline crew scheduling problems with a simple heuristic, we obtain a 48% average decrease in the number of segments of consecutive ones. Furthermore, after reordering the columns, the solution times and the bound values obtained using row split Benders decomposition improve significantly

144

improve. For one of the SP samples, we obtain a better solution time using the row split Benders decomposition than solving the original integer program using CPLEX 6.6 [ILOG 2000].

In general, the convergence rate of the row split Benders decomposition degrades as the number of consecutive ones segments increases.

3.   Reductions In SPP

We have compared known preprocessing reductions for SPPs with known reductions for network flow models extracted from our column split reformulation, and have used the reformulation to establish one-to-one correspondence between reductions heretofore viewed as distinct.

While doing this, we have discovered a new "column split" reduction for SPP. Applying this to a well-studied set of airline crew scheduling SPPs, we identify relations between binary variables that lead to fixing many more of those variables than any preceding work. Moreover, finding so many new reductions may improve the solution time and/or allow some unsolvable real-world SPPs to be solved.

We have also discussed extracting the hidden arcs of the intersection graph of an SPP by probing variables. We note that extending the intersection graph with the addition of new arcs generated by naïve probing may offer new valid inequalities for SPP. However, extracting the hidden arcs of the intersection graph by probing, and finding valid inequalities in the extended intersection graph may be computationally very expensive.

4. **A New Formulation For A Long-Term Aircraft Carrier Scheduling Problem**

We have introduced a new SPP-with-side-constraints formulation for a well-known aircraft carrier deployment scheduling problem. This new formulation significantly reduces problem size, and admits an improvement in time fidelity from months to weeks over a ten-year planning horizon. We use US Navy data for twelve carriers and fixed maintenance periods and show that, if the time resolution is defined in weeks, the percentage of time we can patrol in areas of responsibility increases **1.88% or ten carrier weeks**. Considering the enormous cost of carrier operations and the goal of maximizing such coverage, this is a significant improvement.

B. RECOMMENDATIONS FOR FUTURE RESEARCH

There are several avenues for further research that may prove fruitful.

1. Augmented Lagrangian Penalty Method

For a sufficiently large penalty parameter, the penalty method presented in Chapter III converges to the LP relaxation lower bound. However, if we choose too large a parameter, the penalty method may not converge to the optimal solution because of the ill conditioning we have induced. For certain problems, even if we iteratively increment the penalty parameter value, convergence may not occur. For such problems, we recommend investigating the *augmented Lagrangian penalty method* (e.g., [Bertsekas 1995]). This method is theoretically guaranteed to achieve an exact optimum for finite penalty parameter values.

146

2.   Further Investigation Of The Column Split Reduction

We have only applied the new column split reduction to the initial linear program relaxation, i.e., at the root node of the subsequent branch-and-bound tree.   We recommend applying this reduction periodically during branch-and-bound, and conjecture that it will enhance such reductions at little computational cost.

3.   A Variant Of The Simplex Method

Consider the reformulated problem (*NS*).

minimize $d'y$

(*NS*)   s.t. $\mathcal{N}y = b$

$Sy = 0$

$y \ binary$

Note that the optimal solution to (*NS*) is an extreme point of the following integral polyhedron P:

$$P = \{y \geq 0 \mid \mathcal{N}y = b\}$$

We have sought a restriction of the simplex method that searches for a feasible solution to (*NS*) by moving from one basic feasible solution to another, along the edges of P, always in a cost reducing direction.   That is, we search for an extreme point of P that also satisfies constraints $Sy = 0$.   It requires *exponential time* (i.e., the time needed to implement an algorithm is an exponential function of the length of the input data) to find such an extreme point.   It remains to be seen whether a heuristic search method is worthwhile.

4. Fictitious Play

Consider the reformulated problem (NS). We define the Lagrangian by:

$$L(y,\mu) = \left\{ d'y + \mu(Sy) : \mathcal{N}y = b, y \ \ binary \right\}$$

Consider the following game: player 1 chooses some $y \geq 0$, and player 2 chooses some μ; then, player 1 pays to player 2 the amount $L(y,\mu)$. Player 1 would like to minimize $L(y,\mu)$, while player 2 would like to maximize it.

A pair (y^*,μ^*), with $y^* \geq 0$, is called an *equilibrium* point (or a *saddle* point, or a *Nash equilibrium*) if

$$L(y^*,\mu) \leq L(y^*,\mu^*) \leq L(y,\mu^*), \qquad \forall \ y \geq 0, \text{ and } \mu$$

(Thus we have an equilibrium if no player is able to improve her performance unilaterally modifying her choice.)

We have experimented with *fictitious play* (e.g., [Brown 1951], and [Gass et al. 1996]) to try to solve the game described above. Even if we cannot solve this game, we can still obtain lower or upper bounds for SPP. And, we could not solve this game: The convergence rate of fictitious play is very slow compared to the Lagrangian relaxation.

LIST OF REFERENCES

Ahuja, R.K., Magnanti, T.L. and Orlin, J.B. 1993, *Network Flows*, Prentice Hall, Englewood Cliffs, New Jersey.

Ali, A.I. and Thiagarajan, H. 1989, "A Network Relaxation Base Enumeration Algorithm for Set Partitioning Problem," *European Journal of Operational Research*, 38, pp. 76-89.

Ali, A.I., Han, H.S. and Kennington, J.L. 1995, "Use of Hidden Network Structure in the Set Partitioning Problem," in: *Integer Programming and Combinatorial Optimization*, editors: Balas E., Clausen, J., Springer LNCS 920, pp. 172-184.

Appelgren, L.H. 1969, "A Column Generation Algorithm for a Ship Scheduling Problem," *Transportation Science*, 3, pp. 53-68.

Appelgren, L.H. 1971, "Integer Programming Methods for a Vessel Scheduling Problem," *Transportation Science*, 5, pp. 64-78.

Atamturk, A., Nemhauser, G.L. and Savelsbergh, M.W.P 1995, "A Combined Lagrangian, Linear Programming, and Implication Heuristic for Large-Scale Set Partitioning Problems," *Journal of Heuristics*, 1, pp. 247-259.

Avis, A. 1980, "A Note on Some Computationally Difficult Set Covering Problems," *Mathematical Programming*, 18, pp. 135-143.

Ayik, M. 1998, "Optimal Long-Term Aircraft Carrier Deployment Planning with Synchronous Depot Level Maintenance Scheduling," M.S. Thesis in Operations Research, Naval Postgraduate School, Monterey, California (March).

Baker, K.R. 1976, "Workforce Allocation in Cyclical Scheduling Problems," *Operational Research Quarterly*, 27, pp. 155-167.

Balas, E. and Padberg, M.W. 1976, "Set Partitioning: A Survey," *SIAM Review*, 18, pp. 710-760.

Bartholdi, J.J. III, Orlin, J.B. and Rattlif, H.D. 1980, "Cyclic Scheduling via Integer Programs with Circular Ones," *Operations Research*, 28, pp. 1074-1085.

Bausch, D.O., Brown, G.G. and Ronen D. 1995, "Consolidating and Dispatching Truck Shipments of Mobil Heavy Petroleum Products," *Interfaces*, 25, pp. 1-17.

Bausch, D.O., Brown, G.G. and Ronen D. 1998, "Scheduling Short-Term Marine Transport of Bulk Products," *Maritime Policy Management*, 25, pp. 335-348.

Bazaraa, M.S., Sherali, H.D. and Shetty, C.M. 1993, *Nonlinear Programming Theory and Applications (2nd Ed.)*, John Wiley & Sons, New York.

Beasley, J.E. 1987, "An Algorithm for Set Covering Problems," *European Journal of Operational Research*, 31, pp.85-93.

Beasley, J.E. 1992, "Enhancing an Algorithm for Set Covering Problems," *European Journal of Operational Research*, 58, pp. 293-300.

Beasley, J.E. and Chu, P.C. 1996, "A Genetic Algorithm for the Set Covering Problem," *European Journal of Operational Research*, 94, pp. 392-404.

Benders, J.F. 1962, "Partitioning Procedures for Solving Mixed Variables Programming Problems," *Numerische Mathematik*, 4, pp. 238-252.

Bertsekas, D.P. 1995, *Nonlinear Programming*, Athena Scientific, Belmont, Massachusetts.

Bertsimas, D. and Orlin, J.B. 1991, "A Technique for Speeding up the Solution of the Lagrangian Dual," Working Paper OR 248-91, Operations Research Center, MIT, Cambridge, Massachusetts.

Bertsimas, D. and Tsitsiklis, J.N. 1997, *Introduction to Linear Optimization*, Athena Scientific, Belmont, Massachusetts.

Bradley, G., Brown, G.G. and Graves, G. 1975, *"GNET: A Computer Program for Solution of Capacitated Pure Network Problems,"* Copyright 1975, 1989.

Brown, G.W. 1951, "Iterative Solution of Games by Fictitious Play," in: *Activity Analysis of Production and Allocation*, editor: Koopmans, T.C., John Wiley & Sons, New York, pp. 374-376.

Brown, G.G. and Thomen, D. 1980, "Automatic Identification of Generalized Upper Bounds in Large-Scale Optimization Models," *Management Science*, 26, pp. 1166-1184.

Brown, G.G. and Wright W. 1984, "Automatic Identification of Embedded Network Rows in Large-Scale Optimization Models," *Mathematical Programming*, 29, pp. 41-46.

Brown, G.G., Mcbride, R. and Wood, R.K. 1985, "Extracting Embedded Generalized Networks in Large-Scale Optimization Models," *Mathematical Programming*, 32, pp. 11-31.

Brown, G.G., Graves, G.W. and Ronen, D. 1987, "Scheduling Ocean Transportation of Crude Oil," *Management Science*, 33, pp. 335-346.

Brown, G.G., Graves, G.W. and Honczarenko, M.D. 1987, "Design and Operation of a Multicommodity Production/Distribution System using Primal Goal Decomposition," *Management Science*, 33, pp. 1469-1480.

Brown, G.G., Goodman, C.E. and Wood, R.K. 1990, "Annual Scheduling of Atlantic Fleet Combatants," *Operations Research*, 38, pp. 249-259.

Brown, R.L., Lawphongpanich, S., Looney, R., Schrady, D., Wirtz, J. and Yost, D. 1997, "Forward Engagement Requirements for U.S. Naval Forces: New Analytical Approaches," Technical Report Prepared for Deputy Chief of Naval Operations (Resources, Warfare Requirement and Assessment) N-8, Office of the Chief of Naval Operations, Washington, DC.

Brown, R.L. 1998, Telephone Conversation with the P̲lanning and E̲ngineering for R̲epairs and A̲lterations Activity for the Aircraft Carriers (PERA CV), January.

Brownell, W.S. and Lowerre, J.M. 1976, "Scheduling of Workforce Required for Continuous Operation under Alternative Labor Policies," *Management Science*, 22, pp. 597-605.

Camp, G.D. 1955, "Inequality-Constrained Stationary-Value Problems," *Operations Research*, 3, pp. 548-550.

Caprara, A., Fischetti, M. and Toth, P. 1999, "A Heuristic Method for The Set Covering Problem," *Operations Research*, 7, pp. 730-743.

Charnes, A. and Miller M.H. 1956, "A Model for the Optimal Programming of Railway Freight Train Movements," *Management Science*, 3, pp. 74-92.

Christofides, N. and Korman, S. 1975, "A Computational Survey of Methods for the Set Covering Problem," *Management Science*, 21, pp. 591-599.

Compaq Visual Fortran 1999, A Programming Language, Compaq Computer Corporation, Kingston, New York.

Cook, W., Cunningham, W., Pulleyblank, W. and Schrijver, A. 1998, *Combinatorial Optimization*, John Wiley & Sons, New York.

Crawford, J.L. and Sinclair, G.B. 1977, "Computer Scheduling of Beer Tanker Deliveries," *International Journal of Physical Distribution*, 7, pp. 294-304.

Crowder, H., Johnson, E.L. and Padberg, M. 1983, "Solving Large-Scale Zero-One Linear Programming Problems," *Operations Research*, 31, pp. 803-834.

Dembo, R.S. 1987, "A Primal Truncated Newton Algorithm with Application to Large-Scale Nonlinear Network Optimization," *Mathematical Programming Study*, 31, pp. 43-71.

Department of the Navy 1994, *Naval Doctrine Publication 1 (NDP 1): Naval Warfare*, Naval Doctrine Command, Norfolk, Virginia.

Eben-Chaime, M., Tovey, C.A. and Ammons, J.C. 1996, "Circuit Partitioning via Set Partitioning and Column Generation," *Operations Research*, 44, pp. 65-76.

El-Darzi, E. and Mitra, G. 1990, "Set Covering and Set Partitioning: A Collection of Test Problems," *Omega International Journal of Management Science*, 18, pp. 195-201.

Fiacco, A.V. and McCormick, G.P. 1968, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley & Sons, New York.

Fisher, M.L. 1981, "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science*, 27, pp. 1-18.

Fisher, M.L. 1985, "An Applications Oriented Guide to Lagrangian Relaxation," *Interfaces*, 15, pp. 10-21.

Fourer, R., Gay, D.M. and Kernighan, B.W. 1999, *AMPL: A Modeling Language for Mathematical Programming*, Scientific Press, Danvers, Massachusetts.

Garey, M.R. and Johnson, D.S. 1979, *Computers and Intractability*, W.H. Freeman and Company, New York.

Garfinkel, R.S. and Nemhauser, G.L. 1970, "Optimal Political Districting by Implicit Enumeration Techniques," *Management Science*, 16, pp. B495-B508.

Garfinkel, R.S. and Nemhauser, G.L. 1972, *Integer Programming*, John Wiley & Sons, New York.

Gass, S.I., Zafra, P.M.R. and Qiu, Z. 1996, "Modified Fictitious Play," *Naval Research Logistics*, 43, pp. 995-970.

Geoffrion A.M. 1974, "Lagrangian Relaxation for Integer Programming," *Mathematical Programming Study*, 2, pp. 82-114.

Grossman, T. and Wool, A. 1997, "Computational Experience with Approximation Algorithms for the Set Covering Problem," *European Journal of Operational Research*, 101, pp. 81-92.

Haddadi, S. 1997, "Simple Lagrangian Heuristic for the Set Covering Problem," *European Journal of Operational Research*, 97, pp. 200-204.

Hearn, D.W., Lawphongpanich, S. and Ventura, J.A. 1987, "Restricted Simplicial Decomposition: Computation and Extensions," *Mathematical Programming Study*, 31, pp. 99-118.

Held M. and Karp, R.M. 1970, "The Traveling Salesman and Minimum Spanning Trees," *Operations Research*, 18, pp. 1138-1162.

Held M. and Karp, R.M. 1971, "The Traveling Salesman and Minimum Spanning Trees, Part II," *Mathematical Programming*, 1, pp. 6-25.

Hochbaum, D.S. 1993, "Polynomial and Strongly Polynomial Algorithms for Convex Network Optimization," in: *Network Optimization Problems*, World Scientific Publishing Company, River Edge, New Jersey, editors: Du, D.-Z., Pardalos, P.M., pp. 63-92.

Hoffman, A.J. and Kruskal, J.B. 1956, "Integral Boundary Points of Convex Polyhedra," in: *Linear Inequalities and Related Systems*, editors: Kuhn, H.W., Tucker, A.W., Princeton University Press, Princeton, pp. 223-246.

Hoffman, K.L. and Padberg, M. 1993, "Solving Airline Crew Scheduling Problems by Branch-and-Cut," *Management Science*, 39, pp. 657-682.

ILOG 2000, *ILOG CPLEX 6.6 Reference Manual*, ILOG Ltd., Gentilly Cedex, France.

Lawler, E.L., Lenstra J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B. 1985, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, New York.

Lenstra, J.K. and Rinnooy Kan, A.H.G. 1979, "Computational Complexity of Discrete Optimization Problems," *Annals of Discrete Mathematics*, 4, pp. 121-140.

Marsten, R.E. 1974, "An Algorithm for Large Set Partitioning Problems," *Management Science*, 20, pp. 191-209.

Marsten, R.E. and Shepardson, F. 1980, "A Lagrangean Relaxation Algorithm for the Two-Duty Period Scheduling Problem," *Management Science*, 26, pp. 274-281.

Nemhauser, G.L. and Weber, G.M. 1979, "Optimal Set Partitioning, Matchings and Lagrangian Duality," *Naval Research Logistics*, 26, pp. 553-563.

Nemhauser, G.L. and Wolsey, L.A. 1988, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York.

OPNAV 1990, *Personnel Tempo of Operations,* Instruction 3000.13A, Office of the Chief of Naval Operations, Washington, DC.

OPNAV 1992, *Maintenance Policy for Navy Ships,* Instruction 4700.7J, Office of the Chief of Naval Operations, Washington, DC.

OPNAV 1996a, *Aircraft Availability Schedule,* Report 4710, The Planning and Engineering for Repairs and Alterations Activity for the Aircraft Carriers (PERA CV), Bremerton, Washington.

OPNAV 1996b, *Notional Intervals, Durations, Maintenance Cycles, and Repair Mandays for Depot Level Maintenance Availabilities of U.S. Navy Ships,* Notice 4700, Office of the Chief of Naval Operations, Washington, DC.

OR Library 2000, Set Partitioning Problems, online test data set sited at URL: "*http://mscmga.ms.ic.ac.uk/jeb/orlib/sppinfo.html,*" presented by Beasley, J.E.

Parker, R.G. and Rardin, R.L. 1988, *Discrete Optimization,* Academic Press, San Diego, California.

Pierce, J.F. 1970, "Pattern Scheduling and Matching in Stock Cutting Operations," *Tappi,* 53, pp. 668-678.

Pietrgykowski, T. 1962, "Application of the Steepest Descent Method to Concave Programming," *Proc. Of International Federation Processing Societies Congress (Munich),* North Holland, Amsterdam, pp. 185-189.

Poljak, B.T. 1967, "A General Method of Solving Extremum Problems," *Doklady Akademmi Nauk SSSR* 174 (1), pp. 33-36.

Rockafellar, R.T. 1970, *Convex Analysis,* Princeton University Press, Princeton, New Jersey.

Savelsberg, M.W.P. 1994, "Preprocessing and Probing Techniques for Mixed Integer Programming Problems," *ORSA Journal on Computing,* 6, pp. 445-454.

Schauppner, C.T. 1996, "Optimal Aircraft Carrier Deployment Scheduling," M.S. Thesis in Operations Research, Naval Postgraduate School, Monterey, California (March).

Schrage, L. 1997, *Optimization Modeling with LINDO (5th Ed.),* Brooks/Cole, Pacific Grove, California.

Shapiro, J.F. 1979, *Mathematical Programming: Structures and Algorithms,* John Wiley & Sons, New York.

Stoer, J., and Witzgall, C. 1970, *Convexity and Optimization in Finite Dimensions,* Springer-Verlag, New York.

U.S. Navy 1998, "Carrier: Powerhouse of the fleet," online paper sited at the official website of the U.S. Navy in URL: *"http://www.chifno.navy.mil/navpalib/allhands /ah0197/pg08-09.html,"* The Navy Office of Information, Washington, DC.

Veinott, J.A.F. and Wagner, H.M. 1962, "Optimal Capacity Scheduling-I," *Operations Research*, 19, 4, pp. 518-532.

Wing, V.F. 1986, "SURFSKED an Optimization Aid for Surface Combatant Integer-Deployment Scheduling," M.S. Thesis in Operations Research, Naval Postgraduate School, Monterey, California (September).

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
    8725 John J. Kingman Road, Suite 0944
    Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
    Naval Postgraduate School
    411 Dyer Road
    Monterey, CA 93943-5101

3. Professor Gerald G. Brown 2
    Code OR/Bw
    Naval Postgraduate School
    Monterey, CA 93943-5101

4. Associate Professor Robert F. Dell 1
    Code OR/De
    Naval Postgraduate School
    Monterey, CA 93943-5101

5. Adjunct Professor Wayne P. Hughes 1
    Code OR/Hl
    Naval Postgraduate School
    Monterey, CA 93943-5101

6. Professor Guillermo Owen 1
    Code OR/Bw
    Naval Postgraduate School
    Monterey, CA 93943-5101

7. Professor Richard E. Rosenthal 1
    Code OR/Rl
    Naval Postgraduate School
    Monterey, CA 93943-5101

8. Professor R. Kevin Wood 1
    Code OR/Wd
    Naval Postgraduate School
    Monterey, CA 93943-5101